8·16

The Alternative Atari Newsletter

ATARI

BAUG

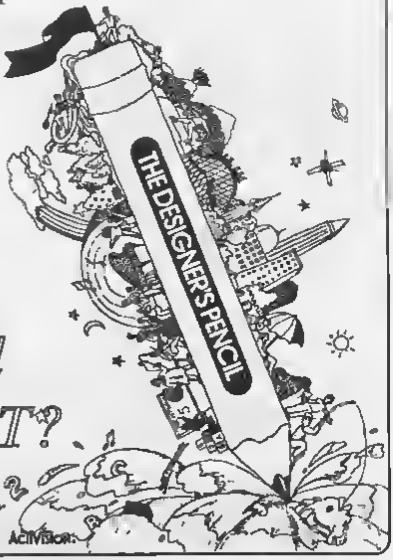
95p | Autumn1989 | Issue 8

Reviewed:
The Designer's Pencil
Keyboard Controlled
Sequencer

Learn to
program in C

Halve the speed of your ST?

With our hardware project - The STopper



THE THIRD ALTERNATIVE MICRO SHOW AND ELECTRONICS FAIR

SATURDAY 11 NOVEMBER 1989
BINGLEY HALL, STAFFORDSHIRE SHOW CENTRE
10 AM - 5 PM

NOW ESTABLISHED AS THE SHOW FOR
THE COMPUTER ENTHUSIAST
THE ALTERNATIVE MICRO SHOW IS MOVING
TO A LARGER SITE AND INTRODUCING
FOR THE FIRST TIME AN ELECTRONICS FAIR

SPECTRUM/BBC/ATARI 8 BIT/QL/EINSTEIN
MEMOTECH/MSX/ADAM/ZX88/LYNX/DRAGON
COMMODORE 8 BIT/JUPITER ACE/TEXAS TI99-4A
ORIC/ENTERPRISE
IN FACT ANY HOME MICRO OTHER THAN ST/AMIGA AND PC's

THE ALTERNATIVE MICRO SHOW AND ELECTRONICS FAIR
WILL BE EXTENSIVELY ADVERTISED IN THE
NATIONAL COMPUTER PRESS, LOCAL RADIO
AND USER GROUP MAGAZINES ETC
AND IS IDEALLY TIMED FOR THE CHRISTMAS TRADE

DON'T DELAY - BOOK TODAY!

Contents

Features

Departments

The Bournemouth and Poole Atari User Group is a member of The Association of Atari User Groups.

Editorial Team

Colin Hunt — D&E Controller
Thomas Holzer — VCS Controller
Glen Lander — Technical Controller - Signews

All contributers are credited next to their articles entry within the contents list.

8:16 is published by the Bournemouth and Poole Atari User Group (BaPAUG).

Letters and articles should be sent to:

8:16
c/o 248 Wimborne Road
Oakdale, Poole, Dorset BH15 3EF

This issue of 8:16 was produced using a Atari 65XE, 1050 disk drive, 520ST SM124 monitor and Apple Macintosh SE. The software used was Mini Office II and TasTalk on the 8 bit, STWriter Gem Kennet and TasTalk on the ST Macrosoft Word, MacPaint, MacDraw Mac Kermit and Aldus Pagemaker Version 3.0 on the Macintosh.

The BaPAUG is a non profit making organisation and no members make money from club activities.

The editorial address for Signews is:

143 Richmond Road,
Leytonstone,
London E11 4BT

Copy date for the next issue of 8:16 is 6th October.
Issue date is 3rd November.

This issue is dedicated to Aaron Richard William Hunt, who weighed in at 8lbs 13oz on Tuesday 22nd August. Our thanks go to Poole General Hospital for all their help. If any future issues of 8:16 are late, you now know why!

Editorial

The Atari Computer Show

Well, the last Atari computer show has been and gone, and for 8 bit users it takes its place as the worst ever, with the number of stands supporting this range being counted on the fingers of one hand. For ST users the show was full of cheap software, new releases, more cheap software, competitions and yes, even more cheap software. The only light shining throughout the show was the Mandarin stand where you could learn about the STOS language system, including all the current extension, and the seminars which proved to be interesting and enjoyable. Maybe if more activites like this were held Atari may of attended? If your an ST user I'll see you at the Computer Shopper Show with all those Amstrad and BBC users!

The Alternative Micro Show

After all the gloom over the lose of the Atari shows Atari 8 bit users should rejoice that the Alternative Micro Show now supports their machine. The last show held in London, contained stands from Computerhouse, Frontier Software and us. The next show being held in Birmingham, will also have Page 6 attending and I believe another user group. If Atari really wish to support all their computers they also should attend. So COME ON ATARI PROVE IT?

Atari 8 bit Users' Convention

You've probably all seen my suggestion that there be an Atari 8 bit Users Convention. Well, the response so far hasn't convinced us that the support for this is large enough, so I offer a challenge. Come and meet us at the Alternative Micro Show and prove that Atari 8 bit users wants convention. We will place a petition on our stand and if enough names sign it at the show then we will stage the event. Can't say fairer than that!

Public Domain Software

I've had several letters asking why we don't run a full PD library, so I thought I'll set the record straight. We originally intended to do this, but dropped the idea for two reasons. One, Page 6 offer an excellent service for 8 bit users and along with several other libraries for ST users. If we ran a library all we could do is offer the same material cheaper. Two, the time spent duplicating disks would mean we wouldn't have any time to write any programs such as these on The 8 Bit Mouse disk. We thus decided to setup a library that contained PD software written by us (the BaPAUG) or modified by us in such a way that the product was improved. This policy has proved very successful and we are now willing to extend this further to allow 8:16 readers to contribute and to include a 'sharewhare' option for programs that we feel are so good the author(s) should gain from there efforts. So if you have written any good software other users may be interested in please send it to us and we will add it to our list and if we it is exceptional we will add to our sharewhare section. For each disk sent in will allow you to select any disk currently within the library. Please note disk must contain your own work or programs modified by you.

Colin Hunt

Subscription rates - annual (4 issues):

U.K.	£3.80
Europe	£8.00
Elsewhere (sea)	£8.00
Elsewhere (air)	£15.00

Commercial advertisement rates:

Full page	£40.00
Half page	£20.00
Quarter page	£10.00

Phone (0202) 677895 for details.

Notice Board

New version of Xtra-RAM

Frontier Software, leading supplier of peripherals and software for the Atari ST range of computers have announced a new upgradable version of their best selling Xtra RAM upgrade for the ST

The new version offers two upgrade options. It will expand any 0 5MB keyboard based ST to 1MB and then when the user wishes it is itself upgradable to give the ST a total of 2 5MB The additional upgrade is performed without soldering by simply changing the RAM chips. It will also expand and 1MB ST to 2 5MB and the Mega 2 ST to 4MB. This extra memory is fully system compatible and can be used with all ST programs including IBM and Macintosh emulators

The Xtra-RAM is user fittable and its installation into an ST does not require any soldering making it one of the few ST upgrades that require no soldering and which do not require the ST to be sent away for the upgrade to be fitted Installation takes about 1 hour and is explained in a fully illustrated 25 page manual

Also included is a floppy disk containing testing software, RAM disk and printer spooler programs. The testing software informs the user of any faults and their possible causes.

The prices for the Xtra RAM are as follows:

Xtra RAM Unpopulated	£69.95
Xtra RAM Upgrade to 1MB	£99.95
Xtra RAM upgrade keyboard ST to 2.5MB or Mega 2 ST to 4 MB	£399 95

Frontier Software,P.O. Box 113, Harrogate, North Yorkshire, England, HG2 0BE
Telephone (0423) 567140/530577

ST Products from Computerhouse

Computerhouse UK, one of the leading suppliers for the 8 bit range of Atari computers have just announced the availability of several ST products.

The first is called Beat Box, a full blown polyphonic song construction set that turns your ST into a drum machine and digital sequencer The system works as a pattern sequencer using sampled sounds. Thirty five sounds are provided with full support for Digisound, ST Replay and Navarone thus enabling you to add your own samples. Also included on the disk is a public domain song player thus enabling you to share your masterpieces. Beat Box costs £19 95

The second is called ST Control. This is a compiled language that can 'drive' any program (GEM or non-GEM) in real time. ST Control costs £59 95

The last is called Switch Back. This is a hardware and software package that enables you to backup your programs and act as a switcher Switch Back costs £69 95

For further details on any of these products contact Computerhouse UK on 01-731 1276 or write to 14 Romaly Court, Landridge Road, Fulham, London SW6 4LL.

Virus Warnings

The Master Sound software being sold at the last Atari Computer Show was infested with the Ghost/Mouse virus. This virus will cause you ST mouse to behave strangely when activated - when you move the mouse down the pointer goes up and, you've guessed it, when you move the mouse up the pointer will go down. It has been suggested by some people that Software Horizons knew about the virus but decided to sell the disk anyway, as this was far cheaper than reproducing them without the virus.

Have you bought any disks from Seven Disk -you know the people that advertise with the slogan 'P.D software at £1 a disk, that I waste money buying blank disks Well, a member of the BaPAUG has recently purchased six disks from this library, only to find four of them infested. This must be a warning to all users who use PD software - check your disks before you use them. Also a note to PD libraries, please, please guarantee your disk against viruses and back the guarantee up by doing your own checking

If you do encounter any virus's on you disks a very good utility to remove them is Virus Killer from Practical. It cost £9.95 and is extremely good value for money

Passing In-Line Parameters To Subroutines.

by Peter Hibbs.

In common with all micro-processors, the Atari's 68000 processor uses subroutines for sections of code which may be repeated more than once within a program. It is often necessary to pass data to a subroutine from the calling program and also for the subroutine to pass data back to the calling program.

There are several methods of passing data to a subroutine. The simplest method is to load the required values into data or address registers and then call the sub-routine which then fetches the data from these registers. This system is useful where the data values are variable and sufficient registers are available but is not ideal where a large amount of data (such as a string of text) is to be transferred.

Another method (which is used extensively by the Atari GEM and BIOS) is to push the data values (or the address of a string of data) onto the stack and then the subroutine has to recover the data from the stack. This method is fairly flexible but is somewhat untidy especially where several strings of data may need to be passed to the routine.

A third method (which does not seem to be used much in Atari software) is to pass the data as in-line data embedded in the calling program immediately after the sub-routine call. This method is only suitable for fixed data, however, since its value is determined at assembly time. This system has the advantage that the data concerned is easily visible at the point in the program where it is used and not in some look up table at the end of the source code. Also the values (such as the string data or addresses) can be easily altered during program development.

This article describes this technique with three examples of subroutines which use in-line data. With the powerful instruction set of the 68000 processor the task of accessing the data in the calling program from the subroutine is quite simple. For example, consider the following code where the data string 'text string',0 is to be passed to the subroutine PRINT which then sends it to the printer port.

```
bsr     print
dc.b    'text string ',0
even
```

The calling program calls the required sub-routine and immediately follows it with the byte constants and terminated with a NUL (0) character. Note that since the string consists of byte size data it could end on an odd address which would cause the processor to generate an 'address' error on return to the calling program. The 'even' pseudo-instruction pads out the string with a NUL character so that the next instruction after the call always starts at an even address.

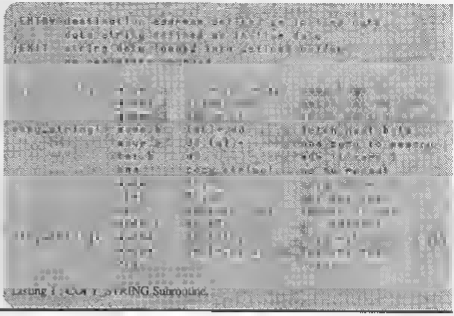
In the subroutine the code would look something like this

```
print  movem.l   a0/d0,-(sp)
       movem.l   6(sp),a0
```

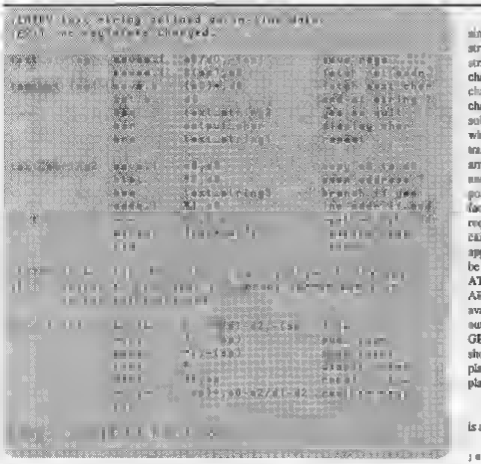
sub-routine code

```
       movem.l   a0,6(sp)
       movem.l   (sp)+,a0/d0
       rts
```

When a subroutine is called the stack pointer is first decremented and then the current return address is pushed onto the stack (4 bytes) with the stack pointer pointing at the last byte saved. On entry to the routine any registers which may be corrupted by the routine are pushed onto the stack (if required). The next instruction (using the 'address register indirect with displacement' addressing mode) copies the return address from the stack into register a0. The displacement value required in this case is 8 since the previous instruction saved 8 bytes (i.e. two longwords) onto the stack. Address register a0 will now be pointing at the first byte of the data string in the calling program code, the sub-routine fetches the data and processes it as required. When all the data has been used, address register a0 must be pointing at the next word after the data string in the calling program code (i.e. the next main program instruction). At the end of the subroutine the address in address register a0 is copied back into the stack using the same displacement instruction so that the return instruction will return program execution to the correct point in the calling program. The saved registers are restored and the RTS instruction returns control to the calling program.



Listing 1 : COUNT_STRING Subroutine.



The following three subroutines demonstrate some practical examples of this technique.

It is often necessary in programs to load a memory store with a text string such as a filename or message. The first example (listing 1) copies a string of data bytes into a specified memory buffer. The calling program specifies the address of the buffer (as one longword) followed by the data string which is terminated with a NUL value. Since all data strings in the Atari operating system must be terminated with a NUL value, this character is also copied into the buffer as well as acting as a terminator.

The format for the call is :-

```
jcall routine
     bsr     copy_string
jaddress of destination
     dc.l    text_buffer
jstring data+NUL
     dc.b    'text string',0
jensure even address
     even
```

The operation of the subroutine is fairly self-explanatory, address register a0 is used as a pointer to the data string, register a1 is used as a pointer to the memory buffer and register d0 holds the data between each fetch and store instruction.

One problem which arises when byte size data is being passed to a subroutine is that the data may end on an odd address. Since the processor can only fetch instructions from even addresses, the return address must be incremented to the next even address if it finishes on an odd address but not changed if it finishes on an even address. When the data transfer has been completed bit 0 of the address register a0 is tested to see if it is a 1 (odd address) or 0 (even address). Unfortunately, the 68000 instruction set does not include a bit test instruction which allows this to be done on an address register so the a0 register is first copied into the data register d0. The least significant bit of d0 is tested and if it is a 0 the routine is terminated after first restoring the saved registers. If the bit is a 1 (an odd address) the a0 register is incremented by one and the routine terminated as before. The 'even' pseudo-instruction to the calling program ensures that the processor fetches the next instruction from the correct address.

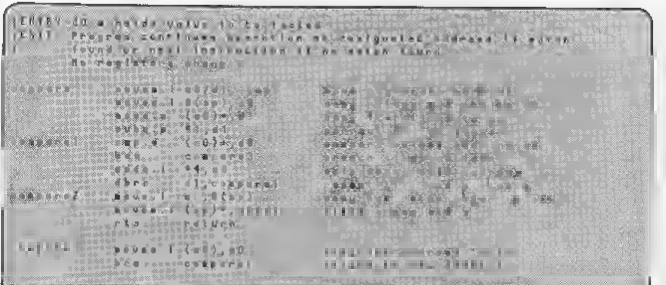
The second example (listing 2) is similar to the first example but displays a string of ASCII data on the screen. The string is again terminated with a NUL character but for this routine the NUL character is not sent to the screen. The text character is sent to the screen at the subsidiary sub-routine 'output_char' which uses the Atari GEMDOS calls via trap #1. This is a VT52 emulator which arranges the screen as columns and rows and automatically increments the cursor position after each call. If the other facilities of the VT52 emulator are required (such as cursor positioning, carriage return, linefeeds, etc) the appropriate ESCAPE or control codes can be embedded in the data string. The ATARI ST INTERNALS book from ABACUS gives a complete list of available control codes. If text is to be outputted to the graphics screen using GEM, the 'output_char' subroutine should be replaced with one which will place the character string at the required place on the graphics screen.

The format for calling the subroutine is as follows :-

```
jcall sub-routine
     bsr     text_string
jtext data+NUL
     dc.b    'text data',0
jensure even address
     even
```

The operation of the routine is very similar to the first example including the address adjustment for a possible odd return address on termination of the routine. The character to be sent to the screen is loaded into d0 before calling the output_char subroutine.

The last example (listing 3) is a subroutine which allows the program to jump to different parts of the program depending on the result of a series of comparison tests. Normally this function would be done with a series of compare instructions followed by a branch-if-equal instruction. The subroutine performs this task using into code in the main program and results in a faster program. The value to be tested is loaded into a data register (d0 is usually convenient) and the number of tests, the values to be tested and the addresses of each destination is passed to the routine as in-line data.



Listing 3 COMPARE.S source

For example, suppose the program calls a keyboard routine and it is required to jump to three different locations if keys A, G or Z are pressed. The value to be compared is first loaded into register d0 (lower word) by the keyboard subroutine and then the compare routine is called. The code will be as follows :-

```
bsr     keyboard        get key value into d0
bsr     escape          call sub-routine
dc.w    3               three tests required
dc.w    'A'             test key A
dc.l    address1        branch address for key A
dc.w    'G'             test key G
dc.l    address2        branch address for key G
dc.w    'Z'             test key Z
dc.l    address3        branch address for key Z
next    instruction     continue program (nosetoh)
```

The keyboard sub-routine (or any equivalent code) moves the value to be tested against the table into the lower word of the d0 register. The compare routine is called and it compares the value in d0 with the first value in the table, i.e. the A. If the value in d0 is also the value A the program will jump to address1, if not, it then tests for values G and Z and again jumps if a match is found. If no match is found the program continues with the instruction after the last address.

The above routines were written with the DEVPAC2 editor/assembler program from HiSoft. There is one point that the programmer should note however. When using the Monitor program to debug the main program the m line data following the subroutine seems to confuse the monitor program when single stepping with the CTRL T option (jumping to the next instruction after a subroutine). When single stepping through a program the CTRL Z option should be used and the sub-routine stepped through otherwise the monitor program will crash.

Mail Order Offers

For the 8 biters :

	RRP	Our Price	Saving
MAC/65 Cartridge by OSS	£49 95	**£39.95**	£10 00

Due to a special agreement with Frontier Software we are able to offer the best assembler for the 8 bit Atari at a £10 discount. Package comes complete with bank switching cartridge and 140 page manual.

For the STers .

Using LOGO On The Atari ST	**£7.95**	**£3.95**	£4 00

This 270 page book by Martin Sims tells you everything you need to know about using LOGO on the Atari ST

To order either or both these items send a cheque (payable to The BaPAUG) or postal order to

8·16 Offers, 245 Wimborne Road, Oakdale, Poole, Dorset

PAC Demo

by Gaven Moran

This demo is primarily an example of how the MOVE command of Turbo Basic can be used to animate player missile graphics. A few other features of Turbo Basic are also used and provide a good example of the way in which turbo basic simplifies programming

What It Does?

An animated 'Pacman' figure is chased across the screen, from right to left, by a ghost. The pacman eats a 'power pill' situated on the left hand side of the screen. At this, the message 'TURBO POWER' appears in large letters across the screen, the ghost changes colour to blue and the pacman chases the ghost back across the screen, eventually eating it.

How It Works?

Lines 230 to 430 contain the set-up procedure. Player / missile graphics are used to represent the pacman, ghost and power pill. The playfield is a GRAPHICS 19 (3+16) screen.

The routine to draw the 'turbo power' message is contained in a separate procedure (lines 190 - 210) and is called by the set up procedure. The TEXT command is used to plot the large letters - these can be quite effective !

Note the use of hexadecimal numbers in line 300. This is a helpful feature of Turbo Basic as it allows direct use of hex numbers in programs - useful for machine language programmers

A DO - LOOP loop is used to read the P/M data into memory. An EXIT command is used to escape from the loop when a -1 is encountered in the data. The data is read into the normally unused memory area between PMBASE and the missiles. A number of animation 'frames' may be stored here (up to 384 bytes in double line resolution or 768 bytes using single line)

Back to the main program which is contained within a DO - LOOP loop which continuously executes the main body of code. Two FOR - NEXT loops control the movements of the players from right to left and left to right respectively. Both these routines call the MOVE-PLAYERS procedure and it is here that the real beauty of the MOVE command becomes apparent.

As I said earlier, frames of player data are stored in the unused part of player memory. The MOVE-PLAYERS procedure uses the MOVE command of Turbo Basic to transfer these into the appropriate player - thus we have a simple but effective player / missile animation system! If this sounds familiar then I recommend you read the article 'PLAYER MISSILE GRAPHICS' in issue 15 of PAGE 6, the technique used there is virtually identical. Who said Turbo Basic had no player/missile commands ?

So there we have it - a simple but effective Turbo Basic demonstration which would probably have taken at least three times as much source code in Atari Basic - not to mention a machine language subroutine. Hopefully the code is easier to read than normal basic too but I had to compress it somewhat to make it fit into one page. Remember, MOVE can be used for other tasks too - such as moving character sets or display lists.

The program listing

```
10 REM PACDEMO by Gaven Moran
20 REM
30 REM 8:16 Issue 8; Autumn 1989
40 --------------
50 EXEC SET_UP
60 DO
70   MOVE PM+55,PG+820,11:POKE HPOSP2,
50:POKE COLP0+$1,72 SETCOLOR 0,0,0 SET
COLOR 1,0,0 PAC=PM:OFFS=12
80   FOR X=200 TO 50 STEP -5:EXEC MOVE
_PLAYERS:NEXT X
90   POKE HPOSP2,0 POKE COLP0+$1,132 S
ETCOLOR 0,4,0:SETCOLOR 1,0,15:PAC=PM+1
1:PAUSE 25
100   FOR X=50 TO 200 STEP 5:EXEC MOVE
_PLAYERS:OFFS=OFFS-0 35 NEXT X:PAUSE 1
00
110 LOOP
120 --------------
130 PROC MOVE_PLAYERS
140   MOVE PAC,PM+570,11:MOVE PG+33,PM
+690,11 PAUSE 7
150   MOVE PM+22,PM+570,11:MOVE PM+44,
PM+690,11:POKE HPOSP0,X POKE HPOSP1,X+
INT(OFFS)
160   PAUSE 7
170 ENDPROC
180 --------------
190 PROC TURBO_LOGO
200   COLOR 1:TEXT 0,3,"TURBO":COLOR 2
:TEXT 0,13,"POWER"
210 ENDPROC
220 --------------
230 PROC SET_UP
240   RAMTOP=PEEK(106)*256
250   RAMTOP=RAMTOP-1024
260   POKE 106,RAMTOP/256
270   GRAPHICS 19:SETCOLOR 0,0,0 SETCO
LOR 1,0,0
280   FOR X=RAMTOP+512 TO RAMTOP+896:P
OKE X,0:NEXT X
290   EXEC TURBO_LOGO
300   PMBASE=$D407 GRACTL=$D01D:SDMCTL
=$022F:GPRIOR=623:COLP0=$02C0:HPOSP0=$
D000:HPOSP1=$D001:HPOSP2=$D002
310   RESTORE *PMDATA
320   OFFSET=%0:PM=RAMTOP
330   DO
340     READ PMBYTE
350     IF PMBYTE=-1 THEN EXIT
360     POKE PM+OFFSET,PMBYTE
370     OFFSET=OFFSET+%1
380   LOOP
390   POKE PMBASE,(RAMTOP/256)
400   POKE GRACTL,3
410   POKE SDMCTL,PEEK(SDMCTL)+12
420   POKE COLP0,240:POKE COLP0+%1,72:
POKE COLP0+%2,15
430 ENDPROC
440 --------------
450 * PMDATA:DATA 60,126,247,127,63,15
,63,127,255,126,60,60,126,230,254,252,
240,252,254 255,126,60
460 DATA 60,126,247,255,255,255,255,25
5,255,126,60 60,126,255,153,221,255,25
5,255,255,111,0
470 DATA 60,126,255,153,187,255,255,25
5,255,85,0,0,0,0,60,126,255,255,126,60
,0,0,-1
```

The STopper by Gary Player

This article describe a very simple piece of hardware that can be used to slow down your ST

Why would you want to slow down your ST?

Well the reason I built it was to cheat at the game 'Alternate Reality' By slowing the ST down it was possible to slow the running attributes so I could choose a good character It has also been useful in gaining an insight into how some graphical games and effects are programmed

How does it work?

One way to slow down the 68000 CPU is to lower the clock frequency but a simpler way is to oscillate the 68000s HALT line. The HALT line is normally used by external peripheral chips to briefly stop the 68000 We will use it to stop and start the 68000 at a high frequency

The 'slowness' is controlled by varying the amount of time the HALT line is held low relative to the amount of time it is held high. For example the following signal will allow the ST to run faster

than this signal.

The circuit

The circuit is shown in Figure 1, and as can be seen it is very simple The 555 IC provides the oscillating signal and the 741S01 provides inversion of the signal and as open collector output to connect to the 68000s HALT line.

The frequency of the signal is given by

$$\frac{1.44}{(R_a+2R_b)C_1}$$

The high time is given by

$$0.693R_bC_1$$

The low time is given by

$$0.693(R_a+R_b)C$$

By varying R_a you will control the amount of time the CPU is stopped. By varying R_b the amount of time the CPU operates and stops is changed. Adjustment of R_a and R_b will depend upon how you wish to slow down your ST eg stops for 1 second and runs for 1 second or perhaps stops for .001 second and runs for .001 second Both slow the ST by 50%, but in different ways.

Construction details

The circuit should be constructed on a small piece of vero-board mounted into a small plastic box The potentiometers are best mounted on the lid and connected to the vero board by wire The power and HALT lines should be connected to a 3-pin DIN chassis socket mounted at one end See Figure 2 for the pinout as viewed looking at the solder tags on the DIN socket The unit will connect to the ST via a straight through three core cable of suitable length via a 3 pin DIN plug fitted to both ends The next stage of the construction involves opening your ST Please note, that any guarantee and warranty your ST has will be invalidated by doing this. Locate the 68000 and attach a wire to the circuit board where pin 17 of the 68000 appears. The power wires can be attached to the power pins of one of the 74 series chips Both these wires and the halt wire should then be attached to a 3-pin DIN chassis socket mounted on the back of the ST Care should be taken to ensure that this socket is wire identical to the one within the unit (see Figure 2) An alternative method of construction is to mount all the circuitry within the ST, but this involves locating 2 potentiometers at a convient place most probably somewhere on top You would also have to add an on/off switch to the halt line in order for your ST to run at its normal speed A list of the major components along with their Maplin order codes is given opposite. The case is a matter of preference and the one listed is only a suggestion

You are now ready to test the STopper Experiment with the pots in order to get accustomed to their effect. Note that you will have to disconnect the unit in order to allow your ST to run at its normal speed.

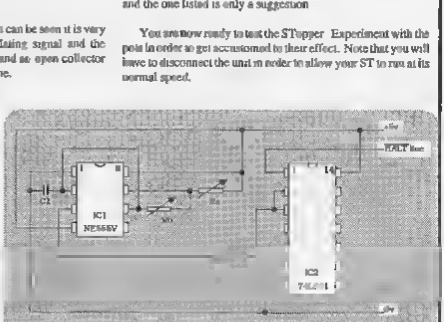


Figure 1 The STopper Circuit Diagram

Figure 2. Pinout of 3 pin DIN chassis mounted socket

WARNING

It has been assumed that anyone who attempts to perform this project has an understanding of soldering, can open up the ST computer without any difficulty and can recognise the different ICs within the ST. If 16 and The BaPAUG can not be held responsible for any damage caused by installing and operating this circuit. At all stages double check you work, and re-check before switching on with the circuit connected. You have been warned.

STopper Items List

Description	Maplin Cat #
C1 Ø Ø22µF ceramic capacitor	BX01B
Ra 22ØKΩ linear rotary potentiometer	FW06G
Rb 1KΩ linear rotary potentiometer	FW00A
IC1 NE555V Timer	QH66W
IC2 74LS01 Quad 2 input NAND gate	YR01B
Vero-board	FL17T
Plastic box (79 x 61 x 40mm)	LH20W
2 x 3 pin DIN chassis socket	HJ13J
2 x 3 pin DIN plug	HH25C
3 core cable	

Tips On Using AtariWriter Plus

Before continuing with part two of this short series it might be a good idea to clear up a point from the first instalment. AWP does not handle proportional text, therefore if you wish print in multiple columns you must turn off your printers proportional fonts. You can achieve multiple columns and proportional text with multiple passes through your printer - more on this in my articles about general word processing later in the series.

Searching For Lower Case Text

To kick off with in part two, did you know that AWP has a strange quirk? It seems that you CAN'T try to search for lower case text! If you press control + shift + letter at the search prompt (START + S) but when the search is initiated, the text can't be found - even if it does indeed exist. The other strange thing about that quirk is that only some letters can be entered, it won t allow you to enter lower case 'A' for example! This doesn't work on all 8-bits, it seems that on some of my associates computers down the computer club they can t enter any lower case text at all to search for. Isn't that weird?

Underlining Spaces

One of the annoying features I find about AWP is the way it underlines text. As you know it won't underline the spaces between underlined words, Antic Magazine recommended that you come out of inverse, type the underline symbol, then go back into inverse again. All that just to underline the spaces between words - very long winded indeed! Yes you guessed it, I have a much better way of doing this. Why bother coming out of inverse in the first place? Just type the underline symbol and then you can use search and replace to replace the underline symbol with an ordinary underline symbol, or better yet, replace it with an inverse space. You CAN replace with inverse text etc. in the replace feature. Good isn't it? On my printer (a Seikosha SP-180 AI) I can underline the underline symbol! Therefore I have to replace the inverse underline. Also an inverse space gives a better quality underline than the underline symbol - try it for yourself.

Adding Graphics

One area of interest for me is the possibility to include graphics within a block of text straight from AWP. This would be especially useful if you want to include a graph in a report as I have to do sometimes for my work. It is possible to pass data straight to the printer using control O + data, and there is a product called Quik-pix that allows AWP to incorporate Print shop icons in text files. Has anybody got any ideas in this area? Perhaps at a later date I can write a program that allows me to plot hi-res graphs and then dump them to a disk, including the correct control O codes, and then print out using AWP? Another possibility is to get AWP to sign my letters at the bottom using the graphics equivalent of NLQ

Widows And Orphans

In the last instalment I mentioned Widows and Orphans. A Widow is a short line at the top of columns, usually the last line in a paragraph. An orphan is a single word or a short line at the bottom of columns. These are to be avoided at all costs, they both detract from the appearance of the page layout by adding white space where it shouldn't be. The quick way to get rid of Orphans is simply to add a return just before the offending paragraph starts. Ways to get rid of Widows includes altering the page length/bottom margin or re-writing the text and constructing it slightly. You should also be aware of the rules for hyphenation. If you must hyphenate a word it must be done between syllables, NEVER in the middle of syllable. You should also make sure that if a word is to be hyphenated, check that the word makes sense afterwards and doesn't give a different meaning!!

Hidden Printer Symbols

Before I sign off, here is a quick tip on getting some 'hidden' symbols from your printer (if it has them). To get the left-hand quote mark type CONTROL + . To get curly brackets type CONTROL + ; for the left-hand pair and type CONTROL + CLEAR for the right-hand side, and on it will not clear your screen.

Introduction To Robotics

Part II Homogeneous Transforms

by S Shannon

1 Introduction

In the first article I outlined some of the general issues in robotics One of the main problems is the geometry of posture. kinematics. To reason about the various links and joints of a robot arm we need some accurate way of specifying exactly where each one is in some precise way This means that we need to be able to attach a co-ordinate frame to each link so that the relationship between any two links can be specified precisely by expressing the general transform between one co-ordinate frame and the other. Ultimately we want a simple way of expressing the relationship between the robot base frame and the end-effector (usually a gripper) In this article, I cover the basic maths of co-ordinate transforms.

2 Representing Posture

The usual choice for a co-ordinate system is cartesian although this is largely a matter of convenience we could you spherical polar or cylindrical systems depending upon the arm geometry. The general transform will be a combination of a rotation and a translation. For brevity let's call this *posture* A translation is easily represented by a 3 vector usually written $\mathbf{t} = (dx, dy, dz)^T$ in text. The vector of displacements is really a column vector which is why there is a superscript ( )T to show that it is transposed. When I refer to a vector by a single symbol like **t** it will be in bold font to distinguish it from a scalar

There are many ways to represent rotations / translations; one of the neatest ways is to use a 4 x 4 transformation matrix called a homogeneous transform.

2.1 Homogeneous Transforms

Let's just plunge straight in. A homogeneous transform between two co-ordinate frames *A* and *B* looks like this:

$$ {}^{A}H_{B} = \begin{pmatrix} R_{3x3} & p_{3x1} \\ f_{1x3} & s_{1x1} \end{pmatrix} = \left\{ \begin{array}{c|c} \text{Rotation Matrix} & \text{Position Vector} \\ \hline \text{Perspective Transform} & \text{Scaling} \end{array} \right\} $$

The top left 3 x 3 submatrix is an orthonormal rotation matrix and the top right 3 x 1 submatrix is a translation vector we can ignore the bottom row of numbers for the moment--suffice it to say that it will be fixed to (0001) for now

2.1.1 Rotations

Imagine that we have two right handed co-ordinate systems *OXYZ* and *O'X'Y'Z'* which are initially coincident, let's rotate the primed system by θ radians about the common Z-axes Now the whole reason we are setting up this formalism is so that we can answer the following question easily: what is the relation between the location of a point p in *OXYZ* and the same point p' measured on *O'X'Y'Z'*? A little bit of geometry tells us that the equations are

$$ \begin{aligned} x &= x' \cos\theta - y' \sin\theta \\ y &= x' \sin\theta + y' \cos\theta \\ z &= z \end{aligned} $$

Or in matrix form,

$$ \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} C & -S & 0 \\ S & C & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} $$

Where $C = \cos\theta$ and $S = \sin\theta$ Even more briefly we could write this as:

$$ \mathbf{p} = \mathbf{R}_{z,\theta}\mathbf{p}' $$

Similarly the rotation matrices for pure rotations about the X and Y axes can be represented by the matrices:

$$ R_{x,\theta} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & C & -S \\ 0 & S & C \end{pmatrix} \qquad R_{y,\theta} = \begin{pmatrix} C & 0 & S \\ 0 & 1 & 0 \\ -S & 0 & C \end{pmatrix} $$

These three matrices are sometimes known as *basic* rotation matrices because they are pure rotations about a co-ordinate system axes and the matrices take on a particularly simple form. How do we represent a more complicated rotation? Say we want to rotate by α radians about the X axis, then rotate by β radians about the new Z axis and finally rotate by θ about the inertial frames Y axis what would the rotation matrix for this look like? Well this is where the hard work starts to pay off. Here is the recipe for any composite rotation.

1. Start with the identity matrix.
2. To rotate about one of the inertial frame's axes *pre-multiply* the result so far by the appropriate basic rotation matrix
3. To rotate about one of the moving frame's axes *post multiply* the result so far by the appropriate basic rotation matrix

Step	Result So Far	Comment
1	I	Start with the identity matrix
2	$R_{x,\alpha}I$	α about X of inertial. premultiply
3	$R_{x,\alpha}IR_{z,\beta}$	β about Z of moving: postmultiply
4	$R_{y,\theta}R_{x,\alpha}IR_{z,\beta}$	θ about Y of inertial. premultiply

The final answer to what initially seemed like a complicated problem is: $R_{y,\theta}R_{x,\alpha}IR_{z,\beta}$

So composite rotations are easy we just do a bit of rotation matrix lego by plugging in the appropriate basic rotation matrix In the appropriate place, multiply out the matrices and we have our answer

2.1.2 Translations

Translations are easily dealt with Again, if we have a point p measured in *OXYZ* and the same point is p' in a translated frame *O'X'Y'Z'*, then what is the relation between p and p'? If we write the translation vector from *OXYZ* to *O'X'Y'Z'* as **t** then the

relationship is just a matter of vector addition

$$\mathbf{p} = \mathbf{p}' + \mathbf{t}$$

2.1.3 Putting Rotations and Translations Together

We can now see that a homogeneous transform is really just a convenient way of writing a rotation followed by a translation. So between frames A and B we can write the transform

$${}^{A}H_{B} = \begin{pmatrix} R & t \\ 0 & 1 \end{pmatrix}$$

2.1.4 Homogeneous Transform Combination and Inversion

Combining a pair of homogeneous transforms is the same as combining a pair of rotations - we can just take matrix products. If we know the homogeneous transforms from A to frame B and we also know the transform from frame B to frame C (written ${}^{A}H_{B}$ and ${}^{B}H_{C}$ respectively) then we may write a transitive chain rule which transforms A to C as

$${}^{A}H_{C} = {}^{A}H_{B}{}^{B}H_{C}$$

Inverting a homogeneous transform is not so difficult as inverting a general 4 x 4 matrix since we can make use of the fact that we have a rotation and a translation. A special property of a rotation matrix is that its inverse is identical to its transpose. If we write the rotation matrix as:

$$R = \begin{pmatrix} n & o & a \end{pmatrix} = \begin{pmatrix} n_x & o_x & a_x \\ n_y & o_y & a_y \\ n_z & o_z & a_z \end{pmatrix}$$

Then the inverse is:

$$R^{-1} = R^{T} = \begin{pmatrix} n_x & n_y & n_z \\ o_x & o_y & o_z \\ a_x & a_y & a_z \end{pmatrix}$$

The translation vector of the inverse is found by "rotation backwards". This means that t_x is projected on to $-\mathbf{n}$, t_y is projected on to $-\mathbf{o}$ and t_z is projected on to $-\mathbf{a}$ by taking scalar products.

So we have

$${}^{A}H_{B} = \begin{pmatrix} n & o & a & t \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} R & t \\ 0_{1\times3} & 1 \end{pmatrix}$$

and its inverse

$$({}^{A}H_{B})^{-1} = {}^{B}H_{A} = \begin{pmatrix} R^{T} & -n.t \\ & -o.t \\ & -a.t \\ 0_{1\times3} & 1 \end{pmatrix}$$

2.2 An Example

If all this seems a bit bewildering, don't worry. You don't need to know how an internal combustion engine works to be able to drive a car. Likewise this example should convince you that you don't really need to know gory details, you just have to be able to follow the recipe of the symbolism.

Let's imagine a simple industrial vision / robot system. Suppose we have a robot arm bolted to one corner of a work surface, and that there is a camera suspended above the surface looking down on to the surface. Both the robot base and camera are fixed and their relative posture are usually known quite well. Say it is known to be:

$${}^{Robot}H_{Camera} = \begin{pmatrix} 0 & 1 & 0 & 500 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 700 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Now, we say that there is an object on the table which has been located by some image processing technique so that we also know the relation between the object and the camera:

$${}^{Object}H_{Camera} = \begin{pmatrix} 0 & 1 & 0 & 100 \\ 1 & 0 & 0 & -100 \\ 0 & 0 & -1 & -500 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Now we want the robot to pick up the object, so the question is, where is the object in the robot's frame? This is easily answered now using the chain rule.

$${}^{Robot}H_{Object} = {}^{Robot}H_{Camera}\,{}^{Camera}H_{Object}$$

$$\Rightarrow \quad {}^{Robot}H_{Object} = {}^{Robot}H_{Camera}\left({}^{Object}H_{Camera}\right)^{-1}$$

So first we need to find ${}^{Camera}H_{Object} = \left({}^{Object}H_{Camera}\right)^{-1}$. Following the transform recipe this is:

$${}^{Camera}H_{Object} \begin{pmatrix} 0 & 1 & 0 & 100 \\ 1 & 0 & 0 & -100 \\ 0 & 0 & -1 & -500 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Plugging in the two matrices, we have:

$${}^{Robot}H_{Object} = \begin{pmatrix} 0 & 1 & 0 & 500 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 700 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 & 1 & 0 & 100 \\ 1 & 0 & 0 & -100 \\ 0 & 0 & -1 & -500 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

So finally

$${}^{Robot}H_{Object} = \begin{pmatrix} 1 & 0 & 0 & 400 \\ 0 & -1 & 0 & -100 \\ 0 & 0 & -1 & 200 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

3 Summary

- Homogeneous transforms are used for representing posture.
- They combine rotation and translation
- They are easy to combine using a transitive chain rule.
- They are easy to invert.

The next instalment will deal with homogeneous transforms as applied to real robot links and joints; we will be able to answer questions like, if I know the joint angles in my shoulder, elbow and wrist then where is my hand?

Introduction to C Programming

By David Watson

C is a language which is currently generating a great deal of interest in the computer world. It has a reputation for being fast and powerful, both of which attract programmers. Unfortunately it also has a reputation for being complex and difficult to use. Although this is true to some extent, C is not necessarily a difficult language to learn, especially if the student has some previous experience with programming languages.

In this article I intend to provide an introduction to C whilst attempting to make life as painless as possible. I will assume that the reader has some previous knowledge of a programming language such as PASCAL or one of the modern structured BASICs. C is not a language for the complete beginner and something like PASCAL is far more suitable for developing good programming habits and understanding the basic concepts.

However before going onto the topic of programming in C, a brief history of programming languages, and of C in particular.

A Short History Lesson.

The first computers to appear consisted mainly of vacuum tubes and plugboards, and were capable of filling whole rooms yet supplied less computing power than a modern day microcomputer such as the ST. Programming languages were unheard of, and programmers would wire up the computer by hand to perform the desired task, perhaps producing a table of sines and cosines.

With the advent of the transistor, computers became reliable enough to be used commercially, and with the development of punch cards and paper tape a large scale back up media was supplied. The first real language to catch on was FORTRAN in 1954, making it now thirty five years old. FORTRAN was developed to handle scientific problems, whilst COBOL which first appeared in 1959 handled the commercial aspects of programming. Both FORTRAN and COBOL have grown in size over the years and are still used because the expense of converting to a different language would be prohibitive.

In the sixties a plethora of languages appeared. Most of them have disappeared over the years with two notable exceptions. FORTH was developed by Charles Moore for controlling radio telescopes and is now used for control programming. It is notable in that all arithmetic is carried out using a stack (i.e. in Reverse Polish Notation). The other language to appear was BASIC. This was an easy language to use and was extremely popular with enthusiasts. Unfortunately there is no standard defining BASIC, only de facto standards. Examples of these standards are Microsoft BASIC for the IBM and BBC BASIC for 8-bit machines.

In 1970, one year after the appearance of BASIC, Nicolas Wirth produced PASCAL. This language is very structured and is therefore ideal for learning with. Unfortunately this is about all it is good for although most PASCAL packages come with extra commands which allow the user to produce graphics and sound with their machine.

In 1979 Nicolas Wirth produced another language, Modula-2. This overcame a lot of the weaknesses present in PASCAL, although it is still very structured and strongly typecast. For those who like the strictness of PASCAL it is probably a good language to move to.

However between the appearances of PASCAL and Modula-2, a third language emerged from Bell Laboratories. This was the brainchild of Brian Kernighan and Dennis Ritchie, and is none other than C. C was written in parallel with the UNIX operating system, which is written entirely in C. Another operating system which has been written in C is GEM. However C is not limited to writing operating systems and programs including text editors, databases, number-crunching programs and some commercially available games have been written in C. For the programmer who likes the high level constructs of structured languages such as REPEAT-UNTIL and WHILE loops and also wants to be able to 'get close to the machine' in terms of low-level programming, C is the ideal choice of language to use.

Compilers And Compilers.

The Atari ST computer programmer is currently faced with a bewildering choice of C compilers and interpreters on the market. There are compilers with GEM-based environments or compilers with command line interfaces, compilers which are very fast and compilers which produce efficient code. Indeed, the only thing which the compilers have in common is a high price tag.

Fortunately though, the public domain can come to the rescue. The Sozobon C compiler is a good compiler and even manages to score over the commercial compilers occasionally. Sozobon C comes complete with all source code (only useful for hard disk owners) and 'dibit', which is a set of routines for performing various tasks (e.g. timers, file operations and so on).

When used with a command line interpreter such as 'Gulam' (which is also in the public domain) it provides a good starting point for C programming, although one meg users without a hard disk will find a half meg RAMdisk useful. Half meg users can use the money saved to purchase a memory expansion.

Finally, a book. If you want to buy a book on C then the best one you can get is "The C programming Language", by Brian Kernighan and Dennis Ritchie. This is an expensive (about twenty five pounds) and thin paperback but is widely regarded as the definitive book on C. If you decide to buy a copy make sure you buy the second edition.

When I first sat down to write this part of the article, I intended to cover the various points of C programming, such as condensed statements, flow control, data types and so on. However, things began to look like a textbook and didn't cover things in as much detail. So I've decided to adopt a different approach.

I will present a program which performs some task, and then explain how it works. The programs will be accompanied with a pseudocode program written in a mixture of English, BASIC and

FASCAL. In this way I hope to explain the basics of C programming whilst avoiding the textbook approach. Whenever I have to make compiler dependent assumptions I will assume the reader is using Sozobon C.

The first routine is really to familiarise you, the reader, with the layout of a C program. It prompts the user to enter a number, and then prints its square (see example 1)

The program is actually held in a function called 'main'. All C programs are collections of functions and the one which is executed first is called 'main'. In this example there is only one function. The body of the function is surrounded by curly brackets (in braces).

The program is commented, and the comment is enclosed within '/*' and '*/'. Comments can go anywhere that a blank line or new line character can go, and are ignored by the compiler. It is not possible to nest comments.

The first line in the program defines the variable 'number' as an integer. The size of an integer varies between compilers, but on the Atari ST it is reasonably safe to assume it is 16 bits long. The statement is terminated with a semicolon. This is different from FASCAL, where the semicolon only separates statements although the only real difference is that with C you must and all statements with a semicolon whilst with FASCAL you can sometimes omit it.

The actual program now starts, and displays a message, asking the user to enter the number to square. The number is then read, and it s square displayed with a suitable message. The task of displaying messages is carried out with the 'printf' command, and that of reading input from the user is achieved with the 'scanf' command. Both of these commands merit further explanation.

The printf command allows the user to format and display a message. This is done by supplying a string which contains format information, followed by any arguments. In addition to ASCII characters, there are two types of special character which can appear in the format string. The first of these supplies format information, and can be called 'backslash' characters. Whenever one of these occurs in the string, output is affected according to Table 1

It is important to realise that two or more characters are used here to represent only one actual character. The \ddd character consists of a backslash followed by up to three octal digits. This allows the programmer to specify the value of the desired character in octal and hence display characters which cannot normally be displayed such as the copyright symbol. If the backslash is followed by anything not appearing above then it is ignored.

Character	Operation
\n	New line
\t	Horizontal tab
\b	Backspace character
\r	Carriage return
\f	Form feed
\'	Single quote
\\	Backslash character
\ddd	Bit pattern
\0	Null character (backslash zero)

Table 1 Printf backslash characters

Substitution	Meaning
%c	Character
%d	Decimal integer
%e	Floating point number (in scientific notation)
%f	Floating point number
%o	Unsigned octal number
%s	Character string
%u	Unsigned decimal
%x	Unsigned hex number
%%	Display % sign

Table 2. Printf substitution characters

The second type of special character indicates that one of the succeeding arguments should be substituted in place of the character. This 'substitution character' provides information on what data type the argument is (e.g. integer, character, etc). and also any format information. The first occurrence of a substitution character is replaced by the second argument of the printf statement, the second substitution character is replaced by the third argument, and so on. A list of substitution characters is shown in Table 2.

That's enough to get you started with the printf statement. If you wish to know more, consult your compiler documentation.

The scanf command supplies a means of allowing the user to enter data, and can be compared to the BASIC 'INPUT' command, or to PASCAL's 'READ' and 'READLN' commands. As with the printf command it needs a format string, followed by the variables to be modified. At this point a problem arises. Because of the way that C passes parameters to functions, it is necessary to pass the address of the variable. This is done by preceding the variable name with an ampersand, & (similar to BBC BASIC). Hence the statement...

```
scanf("%d",&n);
```

is correct, and will place the decimal

Example 1

integer stored into the variable called 'number' (which should be defined as an int). However, if the ampersand is missed out then a row of bombs will appear on the screen when the program is run (try it and see).

This is not checked for by the compiler because if an array is being passed then it is not necessary to supply the ampersand (more on this later). Okay the compiler writers could have checked for this, but then the compiler writers could have done a lot of things differently.

The scanf command is far more complex than this. Unfortunately I'm not going to go into it all just yet - suffice to say, if you want to know more, consult the compiler documentation (yes, I know it's a cop-out, I'm sorry).

One final comment. If the program above is run from within a command line interface such as Gulam, everything will work well. However, if it is run from the desktop then it will finish and the desktop will reappear without giving you time to see the second message appear. One way around this is to duplicate the scanf statement at the end of the program so that it stops and waits for you to enter a second integer before terminating. Alternately, as an experiment, insert the following line after the second printf statement...

```
scanf("\n");
```

When I tried it, the program stopped until I'd typed some characters followed by <RETURN>. This seems a satisfactory way of preventing the program from terminating and immediately returning to the desktop.

One of the failings of the previous program is that it is possible


Example 2

to enter numbers outside a certain range, causing an overflow when the square is calculated. Whereas this would cause an overflow error when using BASIC or PASCAL, with C only the lowest 16 bits are recorded (assuming 16 bit integers). If an overflow occurs then the excess bits are simply ignored, and an invalid answer is obtained. For example, evaluate the square of 5000 using the above program. What we need is to check that the number entered is within some pre-calculated range. Assuming 16 bit integers will give us a range of -32767 to 32767. The largest number we can handle is given by the square root of 32767, which comes out to 181 (truncating the fraction).

The new feature in this program (Example 2) is the 'if' statement. This acts as would be expected...

C statement	BASIC counterpart
`if(condition) do_something;`	`IF condition THEN do_something`
`if(condition) do_something; else do_other_thing;`	`IF condition THEN do_something ELSE do_other_thing`

Note that I've had to split the BASIC if then-else because it wouldn't fit onto one line. If we need to tie several statements to the if/else condition then we do this using the curly brackets in a similar way to the BEGIN END used in PASCAL. For example, we could have coded the conditional part of the program in the following way:-

```
if( number>=-181 && number<=181 ){
  printf("%d",number*number);
  printf(" is the square of ");
  printf("%d\n", number);
}else
  printf("Error  number is outside range\n")
```

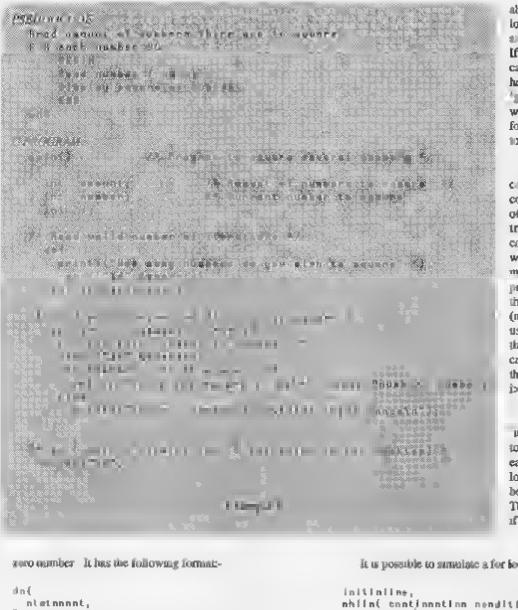
The condition part also has some quirks to trap the unwary, the most notable being the equality operator, '=='. Thus, to test whether or not A is equal to B we use the operation 'A==B', and not A=B (assignment operator). This bug (using the assignment operator instead of the equality operator) is very easy to make and very difficult to track down. The other equality operator (i.e. inequality) is represented with '!='.

The '&&' can be read as AND. This should not be confused with the logical AND which is represented by '&'. Other useful tests are OR, which is represented by '|' (the vertical line is found on the key to the left of the 'Z'), and NOT, which is represented by '!'.

The last improvement to this program which I shall discuss is that of allowing the user to specify how many numbers that he or she needs to square (Example 3).

I now consider the program large enough to comment properly. Commenting a C program is important, as it is very easy to write cryptic code and then be unable to debug it efficiently. Unfortunately, it takes skill and experience to write good comments, and most people don't bother.

There are two looping constructs in this program, a for loop and a do-while loop. The do-while loop performs error checking, and ensures that the amount of loop iterations is a positive, non



above this simply initialises a loop variable to zero, although anything could be initialised. If necessary, you could put a call to an initialisation routine here, although advocates of 'good programming practice' would hardly smile upon you for doing so. It is also possible to leave the field blank.

The second field contains a continuation condition. This condition is checked at the start of each loop iteration. If it is true then the loop will continue, otherwise the loop will terminate. Again this field may be left blank which will produce an infinite loop unless the loop is exited otherwise (more on this later – do NOT use a goto as this will mess up the stack and may cause a crash). In our example above the program will continue until i>=amount.

The final field contains an 'increment'. This can be used to update variables at the end of each loop iteration, such as loop variables. This is what is being done in our example. This field can also be left blank if desired.

zero number. It has the following format:-

```
do{
  statement;
}while( condition is true );
```

The loop terminates when the condition is false. This is an example of a post-test loop - the loop is tested at the end of the loop. Closely associated with the do-while loop is the while loop. This is the pre-test equivalent, and has the following format:-

```
while( condition is true )
  statement;
```

As usual, if there are two or more statements it will be necessary to surround them with curly brackets. The while and do-while loops are essentially the same as their equivalent in other languages, and you should feel quite at home using them. The for loop is different, however, and has the following format:-

```
for(initialisation(sequence);condition (true)
  statement;
```

Thus consists of three separate fields separated by semicolons. The first part contains initialisation information. In our example

It is possible to simulate a for loop with a while loop ..

```
initialise;
while( continuation condition ){
  statement;
  increment;
}
```

You should now be familiar with the layout of a C program. I have also covered flow control in enough depth to allow you to write some elementary programs and hence familiarise yourself with C programming.

In the next two articles I intend to examine arithmetic and logical operations, introducing the different types available to the C programmer and both integer and floating point arithmetic. Functions will be investigated, together with local variables, global variables and parameter passing. Finally, I will cover the topic of pointers. This is going to be a difficult part for me to write as pointers are one of the most misunderstood aspects of C programming, and bugs flourish in programs which involve a lot of pointers. However a firm grasp of the subject is essential to any keen C programmer.

Using DOS 2.5

In case you do not know, DOS 2.5 is one of the disk operating systems available for your Atari 8 bit computer. Other DOSes available include DOS 2.0and DOS 3 from Atari and SpartaDOS from ICD. If you have DOS 2.0 or DOS 3 you should upgrade to 2.5. If you use SpartaDOS then you already have an excellent system which you can use with most of your software. This series of articles is intended for those users who have recently started to use an Atari disk drive or have recently upgraded from DOS 2.0/3 a step I would strongly recommend, and of course for anyone who wants to know how to use DOS 2.5 to its fullest.

The first article (the one you are currently reading) will quickly describe the start up sequence and look at the DOS menu (the disk utility package). Part 2 will cover MEM.SAV, RAMdisks, and those advanced DOS menu commands such as Load Binary File and the error messages. Part 3 will cover using DOS 2.5 from your own BASIC programs with both Atari BASIC and Turbo BASIC being covered. Part 4 will cover the additional utilities available on the DOS 2.5 disk.

Start Up Sequence

In order to use the Atari disk drive with BASIC (or any other environment) you have to perform the following start up sequence:

1. With your disk drive connected to your computer via the serial interface and configured as drive 1 switch on and insert the disk containing the DOS (and only the DOS) you are using.

2a. Switch on your computer without any keys pressed. This will initiate a load sequence from your disk drive. The code being loaded is the resident DOS. That includes all the routines required to read and write to your disks and is called DOS.SYS within the disk directory. At the end of the load sequence you will see the READY prompt.

2b. Switch on your computer with the OPTION key pressed. This will initiate a load sequence from your disk drive. That time, as well as loading the resident DOS the file maintenance software is also loaded (file DUP.SYS within the disk directory). Instead of seeing the READY prompt you will see the DOS menu screen (Figure 1). It is this part of DOS 2.5 that I will look at within this issue.

As well as loading the disk utility package at start up (step 2a above) it is also possible to load it from BASIC using the command DOS. If the file DUP.SYS is on the disk within drive 1 you will be presented with the same menu screen as produced by step 2b above. If you are using a 130XE or memory expanded XL the loading of the DOS menu can be instantaneous by using a RAM disk. We will cover the implementation of this and the use of the resident DOS within a future issue of 8:16.

Dos Menu

In order to execute any option within the menu, all you need to do is type the letter corresponding to your selection followed by RETURN, at the prompt `SELECT ITEM OR RETURN FOR MENU`. If necessary an additional prompt will appear listing the parameters you need to supply before the DOS can perform the function you have chosen.

So lets look at each of the options within the DOS menu.

A. DISK DIRECTORY

This is by far the most used command within the DOS menu. It enables you to read the disk directory on a disk placed within any drive connected to your computer and the RAM disk if present. Upon selection of this command the following prompt will appear:

```
DIRECTORY-SEARCH SPEC,LIST FILE?
```

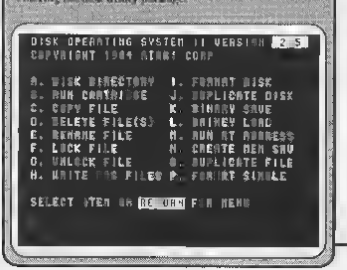
If you immediately press RETURN the disk directory for the disk within drive 1 will be displayed upon the screen. A typical display could look like:

```
   ABS      SYS 037
   DUP      SYS 047
896 FREE SECTORS
```

The number next to each file entry indicates the total number of sectors required to store that file. If the filename is preceded by an asterisk '*' the file is locked (see menu options F and G).

```
* BOB      SYS 037
```

If the filename is enclosed within '< >', the file is unopenable



by DOS 2.0 no part of the file is within a sector numbered greater than 707, the sector limit for DOS 2.0 (see option F)

```
<DOS      SYS>037
```

In order to obtain the directory from other drives you have to enter the following

```
Dn:<RETURN>
```

where 'n' is the number of the drive. It is also possible to be selective by using wild cards, thus

```
D2:* SYS<RETURN>
```

will list all the files with the extender .SYS on drive 2. As well as controlling the drive that the directory is read from it is also possible to control the output device, thus

```
AT1,P:<RETURN>
```

or

```
,P <RETURN>
```

will print the disk directory for drive 1 on the printer (if connected)

B. RUN CARTRIDGE

Selecting this option returns you back to inserted cartridge or internal BASIC on the XL/XE computers. If you had booted with the OPTION key pressed or no cartridge inserted you will get the message NO CARTRIDGE and stay within the DOS menu. If you have MEM.SAV in operation and have entered the DOS menu via the DOS command your program will be copied back into memory

C. COPY FILE

This option allows you to copy files between diskettes within different drives or to make copies of files onto the same disk. When making the selection you will be presented with the following prompt

```
COPY-FROM; TO?
```

You can also use wildcards although there are some constraints. Say we have a file called TEAM.LST and wish to make another copy called TEAMS.LST the following will work

```
TEAM LST,TEAMS LST<RETURN>
```

whereas

```
TEAM *,TEAMS *<RETURN>
```

will fail with the message OPTION NOT ALLOWED

You can also use the command to copy files to the screen or printer

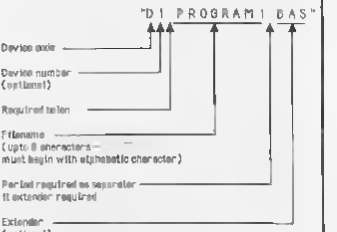
```
TEMP TXT,E <RETURN>
```

or

```
TEMP TXT,P:<RETURN>
```

Filename Structure

The following diagram shows the structure of the filenames used by DOS 2.5. This is the structure used when accesses files from languages such as BASIC. When within the DOS menu the D1: part is optional.



It is also possible to enter data into a disk file by following the following sequence:

```
E:,TEAM.BAT<RETURN>
CHELSEA<RETURN>
@BARNEBUTH<RETURN>
<CNTL>3<RETURN>
```

where <CNTL>3 produces an end of file (EOF) marker and therefore terminates data input. When creating files this way it must be remembered that the data is written to the file when the <CNTL>3 key combination is typed.

There is one last function you can use this option for and that is to append one file onto the end of another..eg

```
TEMP DOT,TEMP DO2/A<RETURN>
```

will append the file TEMP.002 to TEMP.001. You cannot use this command to merge BASIC programs which are stored to disk in a tokenised form .eg using the command SAVE although there are no problems with text files.

Note: The copy command will NOT copy files with the .SYS extender when using a wildcard within the extender, eg D:AUTORUN.* will not pick up D:AUTORUN.SYS. These filenames have to be copied using the full pathname. Do not copy the DOS files to they should be written to disk using the H option

D. DELETE FILE(S)

This option allows you to delete files. Several files can be deleted at the same time on the same drive by using wildcards. When selecting this option the following prompt will appear:

```
DELETE FILE SPEC
```

Under normal operation it is impossible to delete a file without first confirming that you wish the file to be deleted. The

following is an sample session.

```
B:SCROLL *<RETURN>
TYPE "Y" TO DELETE
B:SCROLL ORG?
Y<RETURN>
B:SCROLL LST?
Y<RETURN>
```

It is possible to override this verification by entering the following:

```
B:SCROLL */N<RETURN>
```

When turning off the verification and using wildcards care must be taken not to delete files by accident.

E. RENAME FILE

This option allows you to change the name of one or more files on the same disk. Upon selection of this option the following prompt will appear:

```
RENAME - GIVE OLD NAME, NEW
```

Once again wildcards can be used. The following command

```
TEMP.BAS,*.*<RETURN>
```

will rename the file TEMP.BAS as TEMP.BAS (silly really), whereas the following command

```
TEMP FIL,* TMP<RETURN>
```

will rename the file as TEMP.TMP. All the following commands will produce Error 165 - File name error

```
* TMP,* FIL<RETURN>
* LST,TEMP.LST<RETURN>
* FIL,TEMP *<RETURN>
```

F. LOCK FILE

Locking files is the best way of protecting your files, from accidental erasure, as this option enables you to write protect files, in favour of write protecting whole disks with the disk tabs. When selecting this option the following prompt will appear:

```
WHAT FILE TO LOCK?
```

Once again wildcards can be used, thus

```
D2:* *<RETURN>
```

will lock all files on drive 2. Files that have been locked will appear within a disk directory with an asterisk preceding their name. If you try to delete a locked file with option D you will generate an ERROR 167 - file locked

Wildcards

Wildcards are a means of saving typing time. With DOS 2.5 they can be used to be selective when choosing filenames. The two valid wildcards are a question mark (?) and an asterisk (*). The question mark is used to substitute a single character within the filename while the asterisk can be used as a substitute for several characters. They can also be used within the same filename, eg D1:VER?.* will select all the files that start with VER and one other character within the filename and with any extender.

G. UNLOCK FILE

You use this option to unlock files that have been locked using option F. The prompt is:

```
WHAT FILE TO UNLOCK?
```

As usual wildcards can be used within the filename.

H. WRITE DOS FILES

This option should be used to write the two DOS files to disk you will use during the boot sequence. You do not need these files on every disk that you use. The sequence of events when selecting this option is:

```
DRIVE TO WRITE DOS FILES TO?
1<RETURN>
TYPE "Y" TO WRITE DOS TO DRIVE 1
Y<RETURN>
WRITING NEW DOS FILES
```

If the drive you are trying to write DOS to already has DOS on it the option will be aborted. Upon completion of this option the screen will be cleared and the menu display followed by the prompt SELECT ITEM OR RETURN FOR MENU

I. FORMAT DISK

This is the option you use to format your disks before any data can be written to them. With a 1050 disk drive you will get 1010 sectors (enhanced density), whereas with the 810 you only have 707 sectors (normal density) available for storage. To format a disk in normal density when using a 1050 disk drive use option F. The difference in the number of sectors between the two drives (and DOS 2.0 and 2.5) is due to the fact that the DOS 2.5 / 1050 formats a track with 26 sectors instead of the 18 DOS 2.0 handles. The 1050 disk drive is capable of sensing which disk density the disk currently within the drive is formatted in. When executing this option the following sequence will occur:

```
WHICH DRIVE TO FORMAT?
1<RETURN>
Insert disk into drive 1
TYPE 'Y' TO FORMAT DISK 1
Y<RETURN>
```

If you select the disk directory option after formatting a disk using the 1050 disk drive you will notice that the number of free sectors says 999+ instead of 1010.

Note. DOS 2.0 can only read 707 sectors of the enhanced density disk. To show you any files this will be unreadable by DOS 2.0 the filename will be surrounded by < >(see option A) Also the 810 disk drive cannot read an enhanced density disk.

J. DUPLICATE DISK

This option can be used to create an exact copy of a disk, as long as the disk does not contain any bad sectors. The duplication process is a simple sector by sector copying technique and can be used with a single or dual drive system. When selecting this option the following prompt will appear:

```
DUP DISK-SOURCE,DEST ARCHIVE?
```

By entering

```
1,1<RETURN>
```

a single drive copy will be initiated with the DOS requesting the source and destination disks to be inserted into drive 1

If you have two drives the copying process is more automated as the copy can be done from one disk to another eg

```
1,2<RETURN>
```

will copy from drive 1 to drive 2. This option will also format the destination disk, so there is no need to check the density of the source disk and to pre-format the destination.

K BINARY SAVE

It is very unlikely this this command will be used by beginners as it is intended for the more advanced user. With this option you can save an area of memory to disk as a binary load file. On selecting this option the following prompt will appear:

```
SAVE-DIVE FILE,START,END,(RUN),(INIT,RUN)
```

The minimum requirement for this option is FILE, START, END. The parameters START, END, INIT and RUN are all hexadecimal numbers. I will look at this command in more detail within a future issue of 8.16.

L. BINARY LOAD

This option is used to load any binary file, eg those created with the option K assembler. The prompt for this option is.

```
LOAD FROM WHAT FILE?
```

If the selected file has an INIT and RUN address the file will run immediately unless you use the option /N. I will look at this command in more detail within the next issue of 8.16

M. RUN AT ADDRESS

This option is used to enter the hexadecimal starting address of an object file that has been loaded with the Binary Load option. The prompt for this option is

```
RUN FROM WHAT ADDRESS?
```

As for the last two options I will look at this option in more detail in the next issue of 8.16.

N. CREATE MEM.SAV

This option allows you to create a file called MEM.SAV into which the contents of lower user memory are stored when you execute the command DOS, thus allowing you to leave BASIC and enter DOS and then return to BASIC without erasing your program. The prompt on selection of this option is:

```
TYPE "Y" TO CREATE MEM SAV
```

Some options on the DOS menu allow you to invalidate MEM.SAV, thus giving DOS more room (see example for option O below) I will look at this option and MEM.SAV in general in more detail in the next issue of 8.16.

O DUPLICATE FILE

This option is used to copy files from one disk to another when you have only one disk drive. When selecting this option you will be prompted with the following prompt:

```
NAME OF FILE TO MOVE?
```

While copying files with this option you will be told when to insert the source and destination disk. You will also be informed when each file is being copied. A typical session could look like:

```
NAME OF FILE TO MOVE?
SCROLL BAS<RETURN>
TYPE "Y" TO USE PROGRAM AREA
CAUTION: A "Y" INVALIDATES MEM SAV
Y<RETURN>
INSERT SOURCE DISK, TYPE RETURN
Insert source disk
  <RETURN>
    COPYING - D1:SCROLL BAS
  INSERT DESTINATION DISK, TYPE RETURN
Insert destination disk
  <RETURN>
  ERROR SOURCE DISK, TYPE RETURN
Insert source disk
  <RETURN>
  SELECT ITEM NO RETURN FOR MENU
```

P. FORMAT SINGLE

This option is used to format disks compatible with the 810 disk drive and DOS 2.0. Upon executing this command the following sequence will occur:

```
WHICH DRIVE TO FORMAT?
  1<RETURN>
Insert disk into drive 1
  TYPE 'Y' TO FORMAT DISK 1
  Y<RETURN>
```

That's it for this instalment. If you have any questions or queries about anything related to the use of the Atari disk drives send me a letter, to the usually address and I will endeavour to answer them

The Designer's Pencil

At the recent (and possibly the last) Amiga Show I brought a cartridge for my 8 bit Amstrad called The Designer's Pencil by Activision. I'd heard of the program before but have never seen it on sale locally and at only £2 I couldn't resist it. Unlike other drawing programs where you switch between the control screen, where you select your functions (circle, fill etc) and the drawing screen where you execute it, The Designer's Pencil executes programs on the drawing screen while the control screen is used to construct the program. It is possible to draw freehand, but you have to write a program to do it.

So what use is a drawing package that is in effect, a language designed solely for the purpose of drawing? Is it just a gimmick? Or, does it have a useful purpose?

Well, when I took the cartridge home and finally got round to plugging it in my first reaction was 'A language? Just another gimmick to sell a drawing program.' I was wrong. The Designer's Pencil is a well constructed educational tool which will suit any youngster (or adult) interested in learning to program with immediate results a colourful picture produced by your own program.

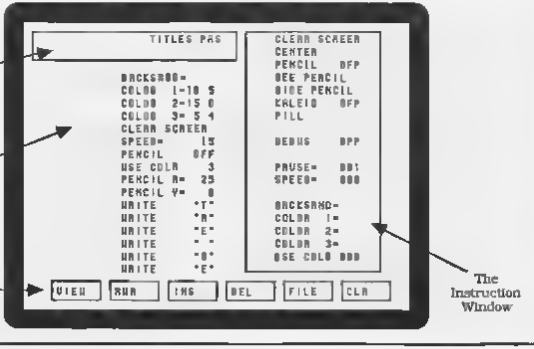
At no stage during the operation of the program do you use the keyboard - thus no typos to worry about. All commands and operations are executed by the use of a joystick plugged into Joystick port 1. All the instructions used by the language (which Activision have called PROG) are selected from the *Instruction Window* located on the right hand side of the screen (see diagram below). By pointing the arrow at the required instruction and pressing the button it is inserted into the program, which is displayed within the *Programming Area*. With some instructions further changes can be made. Once again that is all controlled by the joystick. Take for example, changing the instruction

```
0 = 0     - 000          to      0 = 0     - 010
```

Upon selecting the original instruction it is transferred to your program with the '0 = 0' portion highlighted in blue. By moving the joystick you can change the variable (possible range being A to Z). When you have the required variable pressing the fire button highlights '000' which can also be change (possible range 0 to 255, no negative numbers). Hitting the fire button again completes the insertion of the instruction.

Along the bottom of the command screen are the *Master Commands*. The first two are self explanatory *VIEW* your masterpiece and *RUN* your program. The last one *CLEARs* your program from the *Programming Area*. To protect against accidental usage you are asked to verify the clear, via use of the *Prompt Window*. *INS* is used to insert lines within your program, thus allowing modifications while *DEL* deletes a highlighted line. The last command is FILE. This passes you to the Prompt Window where you can LOAD and SAVE your program to disk and tape, list your program to a printer and save your picture to disk or tape. You can also load in one of eight demo programs stored on the cartridge, which you should study to obtain a better understanding of the language. The train routine is very impressive.

Let me turn this review into a tutorial and examine some of the instructions available. The most useful one must be DEBUG which slows the program down while displaying at the bottom of the screen two lines from the program (the current and next), the current X,Y position of the pencil and the current value of the



current variable. Understanding this two line display will greatly reduce the time taken to debug any mistakes within your program. When the mistake happens pressing the space bar stops the program and returns you to the command screen, with the debug lines still present at the bottom. Another nice feature is that DEBUG can be turned on and off during the program, thus allowing the sections of your routine that are OK to execute at normal speed.

An important aspect of any programming language is the ability to move from the current sequence of instructions, be it to a subroutine or forward/backwards within the current sequence. PROG also allows you to do this by using the instructions: SKIP, JUMP and JSUB

The skip instruction skips the next instruction (simple isn't it), thus within the following routine the instruction B = B - T will not be executed if the statement B=Ø is true

```
SKIP IF B=000
 B = B -    T
```

Two special forms of the skip instruction exist, these being SKIP IF J2= UP, DN, LF or RT and SKIP IF B2 ON These allow you to test the condition of a joystick plug into port 1

The jump instruction allows you to jump to a label. Take for example the following:

```
SKIP IF B>000
JUMP  TO L5
JUMP  TO L6
```

Here the program will jump to L5 if B=Ø and to L6 if B>Ø

The JSUB AT Lnnn instruction allows you to use subroutines for common pieces of code, thus reducing the programs length. Each subroutine has to finish with the RETURN instruction.

Another nice feature of PROG is the ability to add sound to your drawings with the use of NOTE CHn=[B] or RST This instruction plays a note (or rest) for 1/10th of a second in the specified channel (1 to 4), with each channel supporting 3 octaves.

As would be expected for a language written for drawing pictures there are a lot instructions designed to perform specific drawing functions, these include FILL, CIRCLE, RADIUS, ARC FORWARD, BACKWARD, ROTATE, USE COLR and GO (UP, DOWN, RIGHT or LEFT) Most of these commands are straight forward, circle draws a circle with the specified radius, while ARC draws part of a circle. Rotate adds an offset to the direction commands thus allowing you to draw say, spokes of a wheel from the centre out.

Other useful commands are those associated with the use of the pencil, these include PENCIL ON/OFF, SEE/HIDE PENCIL, PENCIL X=000, PENCIL Y=000 SET B=PENC X and SET B=PENC Y When 'on' the pencil will draw while moving around, when 'off' no drawing takes place. With see and hide you can control weather the pencil is seen while drawing

These are some of the commands available, others have been used within the example. Overall, The Designer's Pencil is ideally suited for teaching young children a programming language without overwhelming them with complex instructions, typos and environments. In addition to this it is extremely easy to learn and understand. I would strongly recommend it for any parent, I know my children will learn to program here before moving on to a more complex language.

```
"THE DESIGNER S PENCIL" (C) 1984 ACTIVISION
FILE:DEBB PRG
      BACKGRND= 1 6       Set colours, Colour/Luminosity
      COLOR  T=T0 2
      COLOR  2=14 3
      COLOR  3=10 5
      CLEAR SCREEN
      PENCIL X=   0       Set position of cursor (pencil)
      PENCIL Y=   0
      HIDE PENCIL         Do not show pencil while drawing
L1    KALEID   ON         Draw in all 4 quadrants
      WRITE    "B"        Print text at cursor position
      WRITE    ":"
      WRITE    "1"
      WRITE    "E"
      WRITE    " "
      WRITE    "D"
      WRITE    "E"
      WRITE    "M"
      WRITE    "O"
      KALEID   OFF        Normal drawing
      SEE PENCIL
      SET B=RN  T0        Set B to random number 0-10
      B = B +    5        used as count
L3    PENCIL   OFF        Do not draw while moving pencil
      SET C=RN  50        Random circle size
      C = C +   10
      CENTER              Circles drawn from top therefore
      GO UP    [C]        move pencil to correct position
      SET E=RN   2        Select random colour
      E = E +    1
      USE COLR [E]
      PENCIL   ON         Draw when moving pencil
      CIRCLE R=[C]        Draw circle at with radius=C
      B = B -    1        Decrement count
     SKIP IF B=  0        and exit loop if Ø
     JUMP  TO L3          else jump to label L3
      CENTER              Move pencil to center
      USE COLR   T
      FILL                Fill smallest circle
      HIDE PENCIL
L2    SET B=RN T20        Endless loop which changes
      COLOR  2=[B]        colour 2 randomly
      PAUSE=     2
     JUMP  TO L2
```

Example PROG program.
(Note: You cannot add comments to you program within The Designer's Pencil, the above comments have been added to help you understand the program.)

The Keyboard Controlled Sequencer

An indepth look by Chas Stoddard

In the few years since the ST was released, it has become an indispensable asset to the MIDI musician. Sequencing packages like Steinberg's Pro-24 and, latterly, C-Lab's excellent Notator have allowed people a level of control and sophistication over their music unthinkable of five years ago and the diversity of software for the musician, synthesiser and sample editors, score-writing systems and the like, prove that MIDI is here to stay

Probably the first piece of software that the budding composer will buy is the ubiquitous sequencer. These come in many forms and to suit all pockets, ranging from the simplest of MIDI event recorders to the professional systems mentioned above. The public domain libraries also provide a good source of software including a rather good sequencer. In short, there is something for everyone.

Pro-24 and Notator together represent the 'fashionable' end of the market, in the case of Pro-24, its early appearance guaranteed it wide sales and has proved a most popular package and Notator s combination of sequencer and score-writing system (something Steinberg has now incorporated into their system) is a powerful tool and also has great education value. But there are more ways than one to skin a cat, so it is said, and so this short series of articles, I would like to introduce you to an interesting alternative to the big leaguers, one which I have been using for three years on a variety of projects.

Around the same time as Steinberg were producing Pro-16 for the Commodore 64, a new American software house called Doctor T established itself with a package with the rather verbose title of the Keyboard Controlled Sequencer or KCS for short. Its main interest lay in that it was a "free-form" system, unlike the tape transport approach of other systems where all tracks start together, tracks could be started and stopped at different times or even be transposed by simple one-line commands stored on another track. Whilst requiring a much more disciplined approach to the construction of the piece, you were free to define almost any aspect of its structure. After all not all music is written in popular song formats!

Since then, KCS has grown into what the makers claim to be the most powerful sequencing environment available and I to inclined to agree with there but then I'm biased. As a consequence this article leans heavily in favour of Doctor T s methodology - I make no apology for this as this article is not, strictly speaking, a review but more of me one s observations and appraisal of a system in daily use in a studio

The KCS forms the core of an integrated suite of programs of which up to eight, depending on available memory, can be linked into the KCS via Doctor T's proprietary multi-tasking controller called the Multi Program Environment (MPE), although don't try doing this on anything smaller than a 1040 and, preferably, a Mega 4!

These programs, which can also be run in stand-alone mode, consist of a wide variety of synth editors, a SMPTE read/write package which must rate as the smallest ANSI spec the code device ever, a publishing-quality score-writer which can output to a Linotype photo-typesetter at 2450 dpi a couple of algorithmic composition programs, a real-time graphics editor and two systems which threaten to revolutionise the MIDI world these are a generic synth editor which will handle virtually any synth or MIDI device and a fully fledged BASIC language with links to manipulate directly any type of MIDI data encoded in the KCS More about these later!

Track Mode

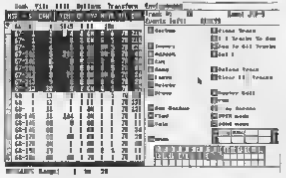
But back to the main reason for this article; what makes KCS such an unusual and interesting beast to use? Well apart from the obvious benefits of the MPE and associated friends, it has a few tricks up its sleeve which make it a very versatile system for the more serious composer or producer At its basic level, KCS operates on three different but quite closely related modes. The first of these, Track Mode, is similar to most of the other offerings on the market in that you have 48 linear tracks to record on. The Track Mode play screen, as well as all the three main edit pages, have GEM menu bars. The Track Mode play screen, in particular has been redesigned. Whilst still showing its ancestry the display is a great improvement; all 48 tracks are shown, there are now six programmable cue points instead of one and the tape transport controls now have fast-forward and rewind options. Underneath the track status display are a series of buttons which access the more immediate functions; to the left are four buttons which enable the recording of controllers and aftertouch MIDI merge and rechannelise. Next along are buttons to switch between track mute and solo, set the quantise value and the exit button to the edit page. In the centre are the transport controls and below them are the real time clock, visual metronome and measure/step displays. To the right appears a column of six buttons which control either the track punch in/out options or functions for Live Edit, which allows the user to employ various keys on the Atari keyboard to modify the data on any track in a variety of ways and the toggles for the six cue point locators.

Virtually all the options in the drop-down menus are duplicated by keyboard shortcuts of which there are quite a few You can merge tracks delete them swap, name tracks, juggle various functions make backups copy and all the other 101 housekeeping utilities commonly found on high-level systems. When it first boots, KCS is in Track Mode Play and is ear into record; playing on the keyboard starts recording on track 1 and continues until F10 or Return is pressed. KCS will continue recording until the end of the current measure is reached whereupon you are dropped onto track 2 and playing commences from the first bar Track 1 sets the length of all the other tracks and you may continue overdubbing in each the same way that you d record parts on a drum machine. When you ve recorded a few tracks it s probably time for a little editing: clicking on Exit drops you into the Edit page for Track Mode.

In previous incarnations, Doctor T has eschewed the GEM environment in favour of his own system of menus. This has led to a somewhat spartan display devoid of the usual system of GEM windows that people normally associate with sequencers. This may well have made people think that it was therefore less powerful than its rivals, an unfortunate perception to say the least, but to cater for those people who prefer a more conventional layout, the KCS now has a hybrid of the two.

The edit page of KCS is split into two; on the left is the data itself which can be edited directly if required by typing in the list and on the other side are most of the editing options arranged in two columns, the less common items are found as the menu bar The advantage of this approach is that all major editing commands are visible at once without recourse to the menu bar; clicking on an item either performs the

Track Mode Play Screen
This display is a great improvement over previous versions, all 48 tracks and their statuses are shown rather than 36 as before.



The Track mode edit screen showing a highlighted range.

Open mode play screen. This shows a piece which uses both sketoric and live performance techniques.

Open mode Edit. This shows the master control sequence for the above.

requested operation or calls up a dialog box for the user to input further information. These dialog boxes are unusual in that, unlike GEM boxes which centre themselves on the screen, Doctor T's boxes, in general, overlay the menu area leaving as much of the edit list visible as possible. Certain menu items do call standard GEM dialog boxes but these are restricted to things like file I/O and one or two functions which were on earlier versions, like Split, but which now have more powerful alternatives or to those house-keeping functions for which there is little point or using a custom window. Also within the edit options are buttons to toggle you to Open or Song modes. Beneath the edit options are three rows of boxes which display the tracks that have been used, clicking on a number takes you to the selected track for editing.

One neat little trick is when a value is needed for a particular field in an edit window, the user merely has to click in that field and then go to a row of seven icons at the bottom of each menu box that allow scrolling up and down through the possible values for that function by units, tens or immediately to the range limits or its centre value. A range of events within the current list can be highlighted by clicking on the first event you wish to process and while holding the button down dragging down the list which will display the selected events in reverse video. All edit functions which expect a range will now have the defined range preset in the appropriate fields as opposed to defaulting to All. The list can be played from the cursor position by pressing the right button while pressing both buttons plays all active tracks from the current cue point – very useful for testing out your edits in context.

Open Mode

Once you are happy with your tracks, you can merge them into a Open Mode sequence, of which there are 126 this is the direct ancestor of the original Commodore 64 version. By muting the tracks you don't want, you can create different mixes. By wiping all tracks you can now start recording all over again. Of course, you might never need to use Open Mode sequences but to do this is to ignore one of the most powerful aspects of the KCS.

While it is perfectly possible to compose an entire piece without leaving Track Mode, most people will prefer to work on smaller sections and this is where Open Mode and Song Mode come into their own. As I've said before, Tracks can be merged into sequences via the All Tracks To Seq option, which will combine all unmuted Tracks into the next available sequence. The sequences are numbered 1 9, A X and 00-92 (sequences Y and Z are reserved by the system for backup purposes) and can be played from the Open Mode Play screen by pressing the appropriate key for the Primary sequences 1-9 and A-X or a two digit number on the numeric keypad for the Secondary sequences. All sequences will repeat forever unless Change Repeats in the edit menu is set to a different value.

OK, so we've got all our sequences into Open Mode, how do we new link all these together into a complete performance? Well you could go to Song Mode and compile a list of sequences to play one after the other – this is the simplest way of doing it – but Open Mode allows you very much greater control over this aspect. KCS has a command language all of its own which allows you to use any spare sequence as a "control" for other sequences.

For example say you have four sequences 1-4 of eight bars each and wish to play them one after the other and then repeat them up a tone. By going to an empty sequence you can enter the sequence number by typing straight into the list in exactly the same way as any MIDI event can be entered. In this case, the column headings at the top of the event display for Note,

A page from the PVG

Velocity and Duration (i.e. the last three columns) take on different meanings. In these three locations go the values for pitch and velocity transposition and the number of times to play like this:

Bar-St	Evnt	Ch	Typ	Note	Vel	Dur
1-1	0		1	0	0	1
9-1	768		2	0	0	1
17-1	768		3	0	0	1
25-1	768		4	0	0	1
33-1	768		1	2	0	1
41-1	768		2	2	0	1
49-1	768		5	2	0	1
57-1	768		4	2	0	1

Assuming a clock rate of 24 pulses per quarter note and a time signature of 4/4 there will be 768 pulses in eight bars hence this number appears in the Time column. The numbers to the extreme left are the bar number and the step within that bar: this is automatically calculated by the system. This is a simple example however, the real power becomes apparent when you can do things like this:

Bar-St	Evnt	Ch	Typ	Note	Vel	Dur
1-1	0		1	0	0	2
1-1	0		3	7	0	1
5-1	384		US	1		
6-1	96		US	1		
6-1	0		PI	*16	-2	
6-25	24		PT	*16	2	
7-1	72	6	PG	107		
9-1	192		4	6	20	1
11-1	192		DX	1		

Here sequences 1 and 3 are started together with sequence 1 set to play twice, four bars later sequence 1 is muted and then unmuted one bar later. At the same time sequence 1 is being unmuted, any active sequence playing on channel 16 is transposed down a tone and a crotchet later is restored to its original pitch. At bar 7, a Program Change is sent out on channel 6 and two bars later sequence 4 starts with all velocity values increased by 20. Finally to bar 11, sequence 1 is shut off.

This ability to use sequences asynchronously is a positive boon: this means that playback structures can be designed which bear no resemblance to conventional forms. Control sequences can activate other control sequences and even be programmed to play sequences at random. If like me you like experimenting with polyrhythmic and polytonal music, you can get up to all sorts of wonderful nonsense with this facility, in fact once this simple command language is learnt and believe me it is simple, making music happen becomes an enjoyable occupation because you are not limited to the facilities that the software's programmers have allowed you; with this system your imagination can really run riot! In addition, the nature of the system makes production work on a wide variety of musical styles simpler because of this orthogonal approach. The system does demand a lot from the person programming it but you gets out what you puts in, as they say!

Of course you can be entirely conventional if you like and simply start sequences one after the other but you can use these control sequences to make your life easier. For example, if you use program changes a lot it can get confusing as to what's going on especially if those changes are buried in amongst your data: altering these can get very frustrating if you need to go fishing around in thirty or forty sequences to find them. With a control sequence you can put all your program changes on one sequence and then call it as the first item in your program.

This makes revoicing a doddle because you can see exactly where things are happening (provided, of course, that you are careful with how you organise your data) and for those synths who are a bit tardy at responding to program changes (hands up all Yamaha owners) you can lay them back a couple of clock steps and give yourself a fighting chance.

Song Mode

I've not discussed Song mode because I never use it. This may seem a bit daft but I find the more generalised environment of Open mode far more useful. This is not to say that Song mode is of no practical use; far from it. Song mode provides a highly organised way of linking sequences together and various methods for modifying the performance in real-time and for the novice user (or indeed for the experienced programmer) but I find that I prefer to control music from Open mode or I can define the 'rules of engagement' myself: anything you can do in Song mode can be achieved in Open mode.

With the KCS, Doctor T's have produced a very interesting piece of software; whilst it incorporates the more standard devices found on other systems, there are a whole range of very powerful editing options not to be found on anywhere else. Most important of these are two innocuous looking items on the menu page called Master Edit and PVG (Programmable Variations Generator). These options take you into a variety of high-level editing functions; Master Edit has six different pages while PVG has ten.

Master Edit

In Master Edit, the user can do things like quantise a track by referencing it to a previously recorded one; this allows you to use a 'groove' track to pull others into the same groove, rather akin to Notator but with a wide range of alternative options to control the transformation, or you can deflam or arpeggiate chords in a variety of ways or 'orchestrate' the notes in the chord out to different MIDI channels. Any continuous controller can be erased, split out to a new track or even thinned out to relieve bottlenecks in the MIDI data, it is invariably possible to thin Mod Wheel or Pitch Bend by a factor of four or five before any noticeable difference to the track occurs. Potentially the most useful tool in the set is the Pitch Map which allows you to change any MIDI note into any other, alter its velocity and change the channel of specific notes. An interesting function is the ability to apply this map recursively, enabling the creation of new source material from a single track.

Programmable Variations Generator

The PVG is possibly the most fascinating area for the composer in that it allows the modification of data or even the generation of new data according to virtually any criteria that you care to name. The interesting wrinkle to this is that all modifications can be performed either deterministically or indeterministically the amount of indeterminism being defined by the user. This environment allows the user to define very complex edits even over specific regions within bars via the Macro facility. A simple example is putting an accent on all events falling within the first eighth note of any bar with the exception of any event on channel 14 that is over a velocity of 120 unless it is

within the pitch range C4 to G5 and its duration is less that 12 clockstops (phew!). With over eight hundred different fields for modifying data, all of which are interactive, the potential is enormous. With this comes the responsibility to ensure that what you program is really what you want, especially when using PVG in an indeterministic manner. Having said that, PVG allows you the freedom to define one-shot utilities that would normally take other systems several lines at the cherry to achieve.

It is interesting to note that other systems have tried to emulate certain aspects of the PVG in their software particularly Steinberg with Version 3 of Pro-24, with varying degrees of success but they fall far short of the sheer flexibility of the PVG. When used to create new material care has to be taken with PVG because it is not optimised to produce music which is musically pleasing to the ear. To be blunt, it is very easy to generate complete garbage but this, of course, depends on your concept of music and how carefully you state your intentions to the system. You might never use certain functions at all (I can think of a couple of functions that I have never found a practical use for) but they are there in case you want them.

On The Road

On a practical level, the KCS is a robust system. Admittedly the learning curve is quite high (what system isn't) but the reward for getting to grips with it is a degree of control over the music that I have never come across in any other product. I've used it as a live playback system for UB40 and Boy George. In the case of UB40, there are two 1040s running simultaneously, one of which is a backup in the event of a failure of the master machine. The computers are controlling a Roland D-550, D-110 and an Akai S900 and are routed via an Akai ME30P II patch bay. Each number in the set is stored as a single Open mode sequence, allowing the whole show to run from one file and eliminating disk loads between numbers.

Both machines are started manually - this seems on the face of it to be a little Heath Robinson but exhaustive trials showed that it is possible to start them within a single clockstep of each other (at 96 ppqn to boot!). In the event of the master machine going AWOL, the patch bay is switched up to the slave machine. Running both machines and flipping between them produces an inaudible glitch. In the 14 months that the band has been on the road, the rig has travelled over 60,000 miles and has performed fruitlessly every night; the slave machine has never been need and the only problem has been a corrupt backup program disk.

In the case of Boy George, the same redundancy was employed with the computers but the MIDI hardware was different. Here the computer was driving a Roland D-110, Yamaha TX-416 and two Samsons SDX samplers each fitted with 8 megabytes of memory and 70 megabyte hard disks. The samplers were used to store backing vocals, brass lines and a few other items; the computers had to cope with over 415K of data, thus stressed the 1040s and the software quite heavily but they managed quite well.

In fact, the KCS was never designed to be used in this fashion but the flexible nature of the system makes it ideally suited to live work. I can remember chuckling to myself when I first saw the Zyklus MIDI performance system, while it was an excellent piece of design, its price was prohibitive and there was virtually nothing it could do that the KCS couldn't also emulate with the appropriate programming.

In The Studio

Here the KCS has proved its worth time and time again. With a variety of support modules linked into the MPE, the KCS becomes a complete MIDI workstation. Voice editing, timecode, scorewriting algorithmic composition and other programs can all be accessed at once. On my Mega 4, I have four TSR programs installed at bootup, five desk accessories, an 800K ram proof ramdisk, 300K of reserved memory for the scorewriter, the KCS and eight other modules running and I still have over 100,000 events left. Most of the MPE modules allow message-passing between tasks; you can play the sequencer from the edit pages in fast synth editor for example and with the imminent release of two systems that I mentioned earlier, the system is poised to take a quantum leap.

These are Orchestrator, which is a generic editor for any synth. As well as the more conventional editing functions which we all nowadays expect to see on such programs there are a number of powerful tools which are very useful; the ability to blend two or more voices together even if they are from different synths; intelligent conversion of voices from one machine to another and the ability to modify any set of parameters in real-time and to record those changes back to the KCS even while it is playing. Those of us who remember the good old days of analog synths and frantic knob twiddling will appreciate the facility to alter the timbre of the voice in such a fashion.

The other is arguably the most fascinating product to arrive on the MIDI scene. This is T-BASIC, a fully-fledged BASIC which resembles ST BASIC (in fact T-BASIC was deliberately designed to be as near a clone of ST BASIC as possible - but without all the bugs!) and which runs under the MPE but with a few important differences.

First there are a number of special functions and keywords which permit the direct manipulation of data within the KCS. There are also functions which affect several critical system variables but, most interestingly, the programmers have designed T-BASIC to auto-compile to object code either when entering instructions from the command line or when loading in from disk. This allows the language to run at a speed that would put most compiled BASICs to shame. In fact, T-BASIC could be used to write any program that you would use other BASICs for. The implication is that for those occasions where there is no direct tool for the job in hand available in KCS T-BASIC can be programmed to do your bidding; an awesome prospect!

In closing, I hope that this article has given you an insight into a different approach to the wonderful world of MIDI. With systems like the KCS and Steinberg's Cubit, the industry is about to take a large step towards the truly integrated working environment. While these systems are very complex, they afford the composer/arranger/producer total control over all aspects of music making. This can only be of benefit to musicians but with this power comes the caveat that all the technology in the world is useless without the creative ideas. When all's said and done, we still need to be musicians first and foremost.

POSTSCRIPT

Chas and his partner Spencer run a company called Earthwatch Productions which specialises in all aspects of MIDI and associated technology. They would be pleased to answer any queries from others Atari users and musicians about MIDI and are also available as MIDI consultants. They can be contacted on 01 378 1799 or by writing to them at the address: Earthwatch Productions 24a Southwark Street LONDON SE1 1TU

Life after an Atari 8 bit?

Last year, in November to be precise, I sold my long lasting friend, my Atari 130XE. I know you might be thinking, "What's the big deal..everyday is selling his 8-bit to upgrade to a ST or Amiga", but I didn't. To be honest, I downgraded (well memory speaking) to the old enemy the Commodore 64. Ah, I hear you say 'things are getting interesting now"; so read on fellas, I am going to tell you the truth.

The C64 is, as we all know, an 8-bitter just like the XL or XE. It comes in a horrible grey case (the '83 version) with a non working power supply unit and a very unreliable datasette. It is best to replace these immediately as this can save you a lot of hassle. For the datasette your best bet is the C16 datasette with an adapter required to plug it into the tape port in the back of the C64. The best bet with the power supply is to buy a unit from a third party manufacturer. The lead from the PSU plugs into the right hand side of the computer, next to the on/off switch and the joystick ports. Bad design this because the lead sticks right out halfway across the desk, looking a bit messy and unprofessional. On my XE the power lead was connected to the back which I think, is much more convenient and looks better (because you don't notice it)

The two joystick ports on the C64 are Atari standard, but the next problem arises when you load a purple title one player game into the machine. Sometimes the joystick routine is written for port 1 and sometimes for port 2. So you leave both joysticks plugged in thinking "hah you can't fool me", load the game and after two minutes of loading it crashes because the joystick in port 1 corrupts the loading procedure (this happens whilst loading Dracowns) Not everything in life is a simple so Kellog Cornflakes, I know!

OK, next subject= The Disk Drive!

The 1541 disk drive is the original drive designed for the C64 and plugs straight in the back port for the drive and / or printer And now the joke: I ve never encountered a computer ware the datasette loads faster than the drive Honest, talking about slow, this drive is slower than a snail in slow motion without a house. Yan B have a better chance watching the floor boards warp Anyway, the drive seems to be reliable, and it has been said that new Commodore drives haves higher failure rate because of their faster speed. Also some software written for the 1541 might not work. So stick to the C64 drives and make yourself a cup of coffee while that mega game loads.

On my XE all I had to load a program was to turn on the drive, insert the program disc and switch on the computer sometimes pressing option as well and bingo it loads. Not so on the C64, you turn on the drive and computer, insert the disk and then you have to type 'LOAD"*",8,1' to load the program, on autoboot here. Mind you this is good for poking into programs.

Next the printer

It's commodores own MPS 801 DMP and its a lot better than the 1029 because it works with programs like Printshop and Newsroom You can also connect an Epson printer via the user port, but very little software supports this and its not easy to find the right cables for it.

And now the most important item, the C64 itself

There is no doubt about it, the best feature on the C64 is the sound chip which is a real synthesiser chip which can do amazing things. Just listen to the intro from the game Savage for example; you get real drums, synths and guitars. All in all, ten minutes of non repeating music and everything crammed into 64K. We all knew about the XE sound chip and a lotta say really

Another good point about the C64 is the usage of 64K, something which should of been done on the XL/XE five years ago. The XE, on the other hand, is a bit faster than the C64 and can handle, for example, vector graphics much better

For word processing I would not recommend the C64 stick to the XL/XE if you can. I think the XL/XE is even better for WP then the ST, why? Because, Page 6 and ANTIC say so, and they should know what they are talking about.

The C64 also has a 3 1/2" disk drive (1581) which is quite fast and an expanding range of software with a lot of American software companies still releasing titles, such as Populous and Battle Chess, on disk. Most of the English software houses are romp, except Ocean and maybe Grumlin, but they do lots of tape games for kids.

Do I regret selling my XE? Well, I have mixed feelings my desk looks a professional look and word processing is a real pain Strip Poker looks better on the XE, but on the C64 I have Mario Whittaker strip Poker so what can I say? If I was to bombard you with technical details about both machines the XE would win, except on sound.

I could go on and tell you so much more about the and things with the C64 but my hands keep sliding of the keyboard so I will sign off with my personnel C64 software top ten (games and music as that's all the C64 is really good for):

1.	Pool Of Radiance	SSI
2.	Rocket Ranger	Cinemaware
3	Grand Prix Circuit	Accolade
4	Zak McKracken	Lucasfilm Games
5	Barbarian II	Palace
6.	Music Studio	Activision
7.	Great Giana Sisters	Rainbow Arts
8.	Katakis	Rainbow Arts
9	Summer Games	Epyx
10.	Battletech	Infocom

Oh yes, people keep asking me if I'm going to upgrade to an ST or Amiga All I'm saying is "the best thing on the Amiga is the Kylie Minogue demo!!"

Thomas Fisher

User Group File

Looking for your local User Group? Then check out the map below to find the nearest group and give its contact a call.

Name: Birmingham User Group
Contact: Wilf Cohen
31 Ram Meadow Yardley Birmingham, B25 8YT
Notes: XL-ST Meetings, BBS

Name: Bournemouth & Poole Atari User Group (BaPAUG)
Contact: Colin Hurst 0202 677895
248 Wimborne Road, Oakdale, Poole, Dorset.
Meetings: 1st Friday every month at the Kinson Community Centre, Pelhams Millhams Lane, Kinson.
Newsletter: 8.16

Name: Cardiff Area ST User Group
Contact: Alan Griffiths
48 Charles Street, Cardiff, CF1 4EF
Notes: ST Meetings

Name: Fast ST Basic Users Group
Contact: Simon Rush
42 York Road, Rayleigh, Essex SS6 8SB
Notes: ST Newsletter

Name: HMS Drake's Computer Club
Contact: John Godfrey
Education Centre, HMS Drake, Plymouth, Devon
Notes: XL-ST Others Meetings

Name: Humberside ST User Group
Contact: Bill Galashan
43 Queensmead, Beverley N Humberside HU17 8PQ
Notes: Meetings

Name: John Player Computer Club
Contact: Robin Smith
7 Trafalgar Road, Beeston Rylands Notts, NG9 1LB
Notes: XL-ST-Others Meetings

Name: London Atari Computer Enthusiast (LACE)
Contact: Glenn Leader
143 Richmond Road, Leytonstone, London E11 4BT

Name: Merseyside Atari Club
Contact: Ken Hall
84 Fitzroy Way Liverpool L6 1JS
Notes: ST, Meetings, Newsletter & BBS

Name: Norwich User Group
Contact: Ken West 0603 661149
45 Colebourn Road, Lakenham, Norwich NR1 2NZ
Meetings: 1st Sunday every month. Contact Ken for time and place.

Notes: The Programmers Advice Line
Contact: 42 York Road, Rayleigh, Essex SS6 8SB
Name: ST Newsletter

Name: Slough Atari Users Club
Contact: Colin Cerne
3 Wardley Close, Stoke Poges, Bucks, SL3 6NS
Notes: XL-ST Meetings

Name: Special Interest Group Atari (SIGATARI)
Contact: Glenn Leader
143 Richmond Road, Leytonstone, London E11 4BT

Name: South West ST User Group
Contact: Karl Fuller
289 Fort Austen Avenue, Crownhill, Plymouth, PL6 5ST
Notes: ST Meetings, Newsletter

Name: Thames Atari Users
Contact: Anthony Cox
48 Stile Hall Gardens, Chiswick, London, W4 3BU
Notes: XL-ST Meetings

Name: Tyne & Wear Atari User Group
Contact: F Gregory
20 Maple Avenue, Silksmead, Sunderland SR3 1QW
Notes: XL-ST Meetings Newsletter

Name: The UK Atari Computer Club
Contact: Roy Smith
PO Box 3 Rayleigh, Essex SS6 8LR
Newsletter: Monitor
Notes: XL-ST No meetings

Name: UK ST Club
Contact: Tony Gosling
Swans Nest, Westerfield, Ipswich, IP6 9AJ
Notes: ST Newsletter

Name: West End Atari User Group
Contact: John Chadwick
208 Jarrot Street, Byker Newcastle-Upon-Tyne
Notes: XL-ST Meetings

Name: User Group Of Ireland
Contact: Mike Casey
3 St. Kevins Park, Kilmacud, Co. Dublin

The Diary

Shows and events that may interest Atari Users:

First Shrewsbury Computer Show
1st to 2nd September 1989
The Music Hall, The Square, Shrewsbury, Shropshire
The Computer Room (0743) 231172

Computer Auction 2nd September 1989
Southgate Technical College, High Street, Southgate, London N14
John Russell & Co. (01) 681 5413

Computer Auction 16th September 1989
Canons Leisure Centre, Madeira Road, Mitcham, Surrey
John Russell & Co. (01) 681 5413

Personal Computer Show
27th September to 1st October 1989
Earls Court, London SW5
Montbuild (01) 486 1951

Computer Auction 30th September 1989
Dacorum College, The Marlowes, Hemel Hempstead, Herts
John Russell & Co. (01) 681 5413

Desktop Publishing Show & Presentations 89
4th to 6th October 1989
London Arena, Limeharbour, London E14
Database Exhibitions (0625) 878888

Computer Auction 14th October 1989
Kempton Park Racecourse, Sunbury on Thames, Middlesex
John Russell & Co. (01) 681 5413

The Third Alternative Micro Show & Electronic Fair
Saturday 11th November 1989
Staffordshire Show Grounds, Weston Road, Stafford
Taurus Computer Systems (0473) 622789

The Computer Shopper Show 24th to 26th November 1989
Alexandra Palace, London
Database Exhibitions (051) 357 2961

Bulletin Boards

Canada

Atari Chamber	604-301-9482
Airport Express	<EX12>404-592-4294
Distillery BBS	<EX12>416-634-7439
DBBS	<EXPR>416-752-1661
L.U.U.T. BBS	<3-12>519-452-5144
PM	<EXPR>416-233-5906
Terminal Island	<EXPR>416-295-6139
The ChaST BBS	<9600>416-274-1225
The Ranch	<EXPR>416-769-5738

Eire

Infotectagon	<300 >0001 764948

Germany (remove 1st 0 and add 010-49)

Charly	<300 >0451-31642

UK

CBABBS	<300 >021-430-3761
Crystal Towers BBS	<3-24>01-836-2813
Gamlingay TBBS	<3-24>0767-50611
Leicester Central	<300 >0533-700914
The City	<3-12>021-553-5486
The Gnome At Home	01-888-8894
The Village	01-464-2516

USA (add 0101 to numbers)

007 BBS	<3 12>804 397 1007
38 Special	<300 >804-473 1872
A.F.A.C.	<3 12>714 781-6774
ACOM	713-669-0603
After Hours	<EXPR>318-747-6255
Andromeda Outpost	703-444-9768
Answering BBS	713-458-9923
Apollo	<3-12>408-246-1432
Arthur's Court	372-858-2823
Atari Base	812-531 5390
Atari BBS #1	<3-12>408-745-5308
Atari BBS #2	<3-12>408 745-5970
Atari BBS #3	<3-12>408-745 2642
Atari BBS #4	<3-12>408 745-4758
Atari BBS #5	<3-12>408 745-5664
Atari Clubhouse	415-736-7478
Atari Connection	851 577 1617
Atari Cockboard	<EX12>201-403 9552
Atari Express	<EX12>404-426-7027
Atari Inn	919 266-6217
Atari Kingdom	<3-12>206-522 3583
Atari Nightline	804-495-8695
Atari Oakland BSS	916-962-2566
Atari Serial Transfer	201-386-5940
Atari Telecom	<EX12>512-494-7083
Atari Backup BBS	<EXST>414-567-3637
Autoconnect	<EX12>417-887-4069
Aviator's Haven	<3-12>804-590-9569
AXE BBS	<EX12>514-696-3506
Bandits Hideout	<EX12>516-643-4963
Bargain Basement	<3-12>502-935 3238
Bargus Cellar	<EXPR>414-265-5149
Barn	701 222 3225
Bastille	651-261-4713
Bates Motel	716-875 7376
Bear Swamp	<3-12>513-644-8714
Bear Tracks II	<3-12>408-968-8072
Bedrock	305 338 1369
Benden Weyr	<3 12>312-975-6003
Big Mac	<EX12>301 574-8130
Billboard BBS	<3 12>803-366-6461
Bit Bashers	<EX12>713-776-3600
Black Hole	<EX12>314-797-8335
Black Water	<EX12>679-279-4176
Blkboards Gallery	<EX12>414-425 7535
Boogie Bass	800-671-6154
Buford Byte Saver	<EX12>404-945-6021
Byte	808-235-4768
C.L.A.M. BBS	<300 >717 254-6085
Cajun Country	<EX12>307 764-3608
Chaos	517 371 1106
Chaser	216-531-2012
Clearing House	<EX12>516-483-7845
CenSTrate BBS	<EXST>301-929-9551
Cabin Command Ctr	216-439-5408
Commodore Atari	<EX12>919-887 2417
CompuClub	<EX12>617-626-0145
CornerSTone	215-775 5684
Country Friends	<EX12>603-485-8580
Country Peddler	313-449-8544
Courthouse ST Expr	<EXST>718 856-1002
Crimson Keep II	<3 12>801-595-8888
CURVE	<3 12>804-288-1430
Cygnus IV	<3-12>305-850-3484
Dark Side	<3 12>804-310-8544
Darkside	<EXPR>615-665-1217
Data-1	808-488-3282
Dead Board BBS	<3 12>717-677-6573
DeadBeat BBS	<EXST>609-853-6084
Deaths Castle	817 231 3447
Disk Bank ST	<9600>305-774-8566
Dragon s Den	<3-24>215-836-3678
Dragon s Lair	516-374-5071
Dynamic Connect	<EXPR>305-863-7202
Eastside Romance	<EXPR>618 259-8618
ECR Products	<EX12>404-379-9230

Baker Street	<3-12>517-546-1309
Bil Oakb BBS	<EX12>309-343-4649
Electronic Age	<3-12>703-620-0851
BK BBS	<EXST>219-288-8722
Enchanted Castle	<EX12>508-769-1800
Enchanted Forest	405-721-6623
Eros	615-378-3250
Experience	907-892-9065
Excalibur	<3-24>303-776-1411
Exodus BBS	<EX12>508-655-5229
Express	<EX12>314-225-8710
Flights Of Fancy	<300 >402-244 5939
Forbidden Zone	415-665-1411
Forum	301 552-2573
Four Acres OnLine	<EXPR>415-733-3192
Four Aces	<EXPR>403-278-4215
Ft. Nausea	<300 >216-494-9355
G.R.A.S.P BBS	<EX12>804-746-8822
Galactica BBS	<300 >804 273-0971
Garfield BBS	<EXPR>716 873-4232
Gateway	314-647-5290
Gnomes BBS	302-994-8779
Gorilla BBS	<3-12>716-225-4187
GraveYard	<EX12>516-785 7034
Griffins Nest	<EX12>402-466-5399
H.A.C.K	713-583-8900
Hackers Systems	<EX12>914-693 2401
Hall Of Justice	<EXPR>214-790-2512
Hall of Immortals	<EX12>301-242-2588
Harv Talbitz	<3-12>501-847-8593
High Sierra BBS	<3-12>702 356-1418
Hole In The Wall	<EX12>907 243-3522
Horizon BBS	<EXPR>216-641-9704
Honored Society	<EX12>618 544-7533
HQ	<3-12>915 334-5897
I.R.S	<3-12>803 366-5606
ICD	<EXST>815-968 2229
Imagination BBS	<EX12>513 787 3777
Inside The 8-Bit	<EX12>404-718-2474
Iron	316-683 1489
Jolly Baby	918-622 1121
JunkYard	<EX12>516-253-0079
KC Express	<EXPR>816-765 1150
KSRYGMA BBS	<3-12>404-672-9908
Klingon Tactical Command	614-866-4733
L'Imperial	<EXPR>415 328-6340
Lake Shore Area	<EX12>312-348-2713
Lamp Post	<EXPR>717-854-5107
Le Chateau	<EXPR>404-297-8975
Level 42	<EXST>309 387-6456
Liberty Bell Expr	<EXPR>504 783 3583
Lost Temple	<EXPR>206-235 0617
Luna City	<3-12>703-455-0040
M.A.C.E	<3-12>307 784-0341
M.O.U.S.E	<EX12>219-674-9288
M.T.V	216-381-9952
Maelstrom	512-697 8331
Magicians Footnote	301 233-5360
Magnathea	<EX12>602 833-9216
Mainstreet Muster	<3-24>804 271-3298
Master Yankee	<3-24>801-942-0034
Matterhorn Monitor	<EXPR>717 896-3161
Maximum Security	<3-12>808-422 2247
Mid-Ohio A.U.G	419-756-5075
Middle Earth	<3-24>515 224-6557
Midnight Express	<EXST>804-379-4156
Midnite Madness	<3-12>512 523 1636
Mister Message	<EX12>516-454-7691
MODEM LINK BBS	<EX12>618 345-1116
MODEM BBS	<EXPR>716-381-5139
Monsoon Hotel	<EXPR>602-881-5868
Mountain Aerie	<300 >307 234-2411
Movie Theater	301-868-5011
Mr.Robinson's	<EXPR>404-979-8456
MVAFIX	315-732-8591
MV HOUSE	<EX12>805-929-1624
Nassau County BBS	516-565-3932
Never Ending Sky	<3-12>716-586-4854
Night Shift Atari	<300 >616-641 5034
No Atlanta	<EXPR>301-632-3671
O.R. (atari only)	<3-12>215-779-6771
Omni BBS (no more)	<3-12>505-757-9532
Of Harbor's BBS	<EX12>516-884-4140
Omega Base	<EX12>301-384-2966
Outer Limits	612-871-2384
Pac	914-887-4306
PARANOIA CENTRAL	<EXPR>907-253-6445
Part-Time BBS	<EX12>216-582-1196
Party Place	<3-12>516-791-4556
PERKY PENGUIN	<EXPR>716-671-1449
Paramount City	<EXPR>504-687-1306
Phoenix BBS	201-335 5274
Pilot	704-865 7657
Pirate Ship II	<3-12>815-238-8531
Power Station	808-944-0639
Prairie Chip	<3-24>507-655-4152
PRO BOARD	<EX12>512-467-4004
Project Atari	<3-12>317-852-6608
PSL	<3-12>404 794 1924
PYTHON	<EX12>312-680-5106
R.A.S.S. BBS	<EXPR>503 746-9637
RACE Rendezvous	<EXPR>406-543-8118
Rampage	404-923 1222
Rat's Nest	<EXPR>219-457 3777
RatCom Express	<EX12>301-457 1813
Ravenloft	703-644-5734
Rebel BBS	<3-12>702-435-0786
Red Desk	317-924-6286
Reflex Point	808-531-2895
RIACE ST EXPRESS	<EXST>401 521-4234
Rich Honeymoon	<3-12>609-927-6678
Road House	<EX12>203 278-5583
Rock'n Roll Harbour	305-821 2237
ROUND TABLE	<EXST>517 546-1309
S.A.C	<EXPR>201 326-1776
S.S.T. BBS	<3-24>702-645-6185
SAC Base BBS	<EX12>815-744-6501
Sanctuary BBS	412-343-1422
Schoolhouse	504-241-5673
Schrodinger's Cat	<EXPR>804-490-9866
Service BBS	804-495-9541
Shadow Kingdom	<EX12>907-479-4524
Shak	<EX12>512-670-1628
Sierra Express	<EX12>702-356-1418
Small Wonder BBS	502-475-5493
Snare BBS	702-438-2208
Snake Pit	<EX12>215 776-7856
South Of The Border	516-421-5489
SouthEast AMIS	<3-12>704-541 3306
Southern Comfort	516-242-1187
Space Station 69	<EX12>607 754-8016
Spider's Web	<EX12>203-445-4094
Stallion's Corner	<3-12>419-428-0719
Star Trek III	516-566-0589
StarBase	<EX12>412-423-4535
Starbase 18	215 279-8722
Summit Gen (ST)	217-267-3530
Starship	619-726-3491
Startup BBS	<EXPR>716-885-0452
STate of the Art	<EXST>516-791-4556
Stop N Swap	216-231-3034
Style Express	<EX12>305-282-8746
SUNDOWN SPECIAL	<EXPR>716-544-5734
Super Support	503-926-1980
SurfBoard	<EX12>604-339-3915
Swap Shop	505 878 1422
Synchronicity	703-522-7782
T.A.P BBS	<EX12>216 228 7335
T.A.R.C	918-664-0459
Tattletale BBS	<EX12>315-685 5585
THE PUBLIC DOMAIN	<EX12>907-338-5005
THE ROUND TABLE	804-276-7368
THE TREASURE CHEST	<EX12>216-455-0604
THE "WQNR" BBS	<EX12>516-691-7456
The 25th Century	<3-96>703-898-4537
The Alcove	<3-24>804-285-2607
The Atari Vault	504-737-8212
The Balloon Works	419 289-8392
The Bear's Den	<300 >716-731-4166
The South	<EX12>805-965 5709
The Bogus Zone	206-833-2455
The Bookstore	<EX12>213-921-0065
The Boot Factory	<3-96>804-262-9289
The Crypt II	<EX12>305-581-9427
The Duck Crypt	<EXPR>404-968-4388
The Data Connection	301-393-5027
The Dead Board	<3-12>717-477-9575
The Dragon's Lair	<3-12>219-295-1308
The Embassy BBS	504-392-0200
The Empire	805-988-0106
The Enterprise	<EX12>718-745-0068
The Farm	<EX12>315 736-3920
The Frontier	<EXPR>215-777-5123
The Great Unknown	601-833-3646
The Harlem Guild	<EX12>301-731-1133
The Haunted Forest	<3-12>404-231-4584
The Hawk BBS	<EX12>716 297-8596
The Help	316-683 7514
The Keep BBS	<EX12>717-675-4068
The MadHouse BBS	<EXPR>418-865-7572
The Missing Link	<3-12>606-271-1466
The New Church	<3-12>602 848-0280
The Phoenix	<3-12>812-466-3478
The Poor House	<EX12>317 362 3352
The Reef	2400-1200<EXST>306 848 3371
The Repair Shop	<EX12>801-967-8738
The Rainpost	<3-12>804-732-0923
The SMURF's Turf	<EX12>718 526-5395
The StormBringer	<EX12>206-897-5496
The T.A.L.K	<3-12>313 326-4126
The Time Zone	<EX12>401 728 5365
The Tonight Show	712-323 1216
The Trading Post	617 963-8138
The Trap Door	<300 >513-979-4041
The Ukranian	<2400>516-292-9747
The Unicorn	305-729-9575
The Write Number	<EXST>915 751-7833
The Wrong Number	<EXST>915-751 3306
The Zombie	617-994-6797
Thieves Guild	607 729-2141
Thieves World	919-596-0581
Third World	404-751-8498
Town Calor	808-235 5538
Truffle BBS	<EX12>516-727-6178
Twilight BBS	<3-12>215 536-3818
Twilight Zone ST	<3-12>502-456-4408
Ups And Downs	<EX12>401-683 5961
USS ENTERPRIDE	<EXST>213-929-9449
W.O.P.R	<EXPR>512-467 7554
Webbed Sphinx	512-299-3665
Westar	<EX12>704-758-5285
Wet N' Wild	516-421-4161
Whopper	801-755-7052
Wicked Good	<EX12>207 774-5045
Wise Byte	<EX12>516-349-7835
Wizard's Castle ST	<9600>402-435 1426
Wonderful World Oz	<EXST>800-423-3140
Wood Shop BBS	<EXPR>304-232-1564
World of Oz	<EX12>806-423-2754

Key

<EXST>	Express ST
<EX12>	Express 1200
<EXPR>	Express 1030
<300 >	300 Baud
<3-12>	300 or 1200 Baud
<3-24>	300 to 2400 Baud
<3-96>	300 to 9600 Baud
<9600>	9600 Baud

Note: The bulletin boards that make up this list were taken from Page 6 and downloaded from the ICD BBS in the states. 8:16 makes no guarantee to the accuracy of any of the telephone numbers listed.

Star Printers

LC-10 B/W ---------- £165.00
LC-10c Colour ---------- £199.00
LC-24/10 24 Pin ---------- £255.00

p&p + ins £4.00

Limited Offer Printshop only £10 if purchased with printer

256-12ns
RAM
CHIPS
£4.75 each

Vertical Disk Rack £6.25

50 Disk 5.25" Storage Box £7.95

2 metre 25 way D'Type 'Easy Link' Printer Cables £4.95

5.25' DS-QD 96 tpi disks with write protect tabs/labels
box of 10 £3.75
box of 100 £32.95

320K RAM upgrade kit for 130XE £59.95

128K RAM upgrade kit for 65XE £34.95

320K RAM upgrade kit for 65XE £69.95
(kits include all ram chips)

NEW

£159.95

"System Solver"

Computer Desk Suite

inc chair

Ideal for home or business

Write in for more details on this superb space saving suite!

NEW

Continuous Stationary Envelopes

5" x 11" self seal type

£10 per 100

Atari XEP-80 'True crisp 80 column screen module with built in standard printer interface £47.95
Turbofile 80 for the XEP-80 module £29.95
Family Finance £12.95
Printshop disk £14.95
Printshop Icon disk £5.00
Visicalc spreadsheet £16.95
B-Graph stats/graphs £15.95

Alternative Reality (City)	£15.95	Fernie Grenadier	£18.00
Arkanoid	£5.95	Questron	£15.00
Atari Planetarium	£15.00	Racing Destruction Set	£8.95
Battlecruiser (SSI)	£20.00	Silent Service	£8.50
Beachhead II	£6.95	Summer Games	£6.95
Bruce Lee	£6.95	Tigers In The Snow	£9.00
Conan The Barbarian	£6.95	Ultima III	£16.95
Fight Night	£6.95	Warship (SSI)	£16.95
Leader Board	£6.50	Whirlinurds	£6.95
Orgs (Orgas)	£15.00	Wizard Of Wor	£5.95
		Zorro	£6.95

SHAREWARE

G.O.S. (Graphics Disk Operating System) £2.95
MYDOS 4.50 ins. tech notes and disk fixer utility £3.95
P:15 Counterstrem Set disk £3.50
AMODEM 7.50 (split baud handler) £3.50
C.D.T.C. (Cartridge / Disk / Tape copier tools) £3.50

SpartaDOS X £46.00

14, Romily Court,
Landridge Road, Fulham,
London SW6 4LL
Telephone 01-731 1276

ACCESS - MASTERCARD - EUROCARD ACCEPTED